亚马逊云科技

用户指南

Amazon Identity and Access Management

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Identity and Access Management: 用户指南

Copyright © 2026 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon 的商标和商业外观不得用于任何非 Amazon 的商品或服务，也不得以任何可能引起客户混淆、贬低或诋毁 Amazon 的方式使用。所有非 Amazon 拥有的其他商标均为各自所有者的财产，这些所有者可能附属于 Amazon、与 Amazon 有关联或由 Amazon 赞助，也可能不是如此。

Amazon Web Services 文档中描述的 Amazon Web Services 服务或功能可能因区域而异。要查看适用于中国区域的差异，请参阅 中国的 Amazon Web Services 服务入门 (PDF)。

Table of Contents

什么是 IAM？ 1
为什么应该使用 IAM？ 2
对您 Amazon Web Services 账户 的共享访问权限 2
精细权限 3
在 Amazon EC2 上运行的应用程序针对 Amazon 资源的安全访问权限 3
多重身份验证 (MFA) 3
联合身份 3
实现保证的身份信息 3
PCI DSS 合规性 3
何时使用 IAM？ 4
当您执行不同的工作职能时 4
当您获得授权访问 Amazon 资源时 4
当您以 IAM 用户身份登录时 5
当您代入 IAM 角色时 5
当您创建策略和权限时 5
我如何管理 IAM？ 6
使用 Amazon Web Services 管理控制台 6
Amazon 命令行工具 7
使用 Amazon SDK 9
使用 IAM Query API 9
IAM 的工作原理 9
请求的组成部分 11
如何对主体进行身份验证 12
授权和权限策略基础知识 12
..... 13
比较 IAM 身份和凭证 14
术语 14
IAM 用户和 IAM Identity Center 中用户之间的区别 16
现有身份源的联合用户 17
提供用户访问权限的不同方法 18
支持编程用户访问 20
权限和策略如何提供访问管理 22
策略和账户 22
策略和用户 22

策略和 IAM 组 23
联合用户会话和角色 23
基于身份和基于资源的策略 23
使用 ABAC 授权定义权限 24
ABAC 与传统 RBAC 模型的比较 25
入门 27
保护 IAM 用户 27
查看您的 Amazon Web Services 账户 ID 27
查看您的 Amazon Web Services 账户 ID 28
为您的 Amazon Web Services 账户 ID 使用别名 29
创建账户别名 30
删除账户别名 30
规划对 Amazon 账户的访问权限 32
IAM 用户的使用案例 32
将多重身份验证用于身份验证 44
安全最佳实践和使用案例 46
安全最佳实践 46
要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon 47
要求工作负载使用具有 IAM 角色的临时凭证访问 Amazon 47
需要多重身份验证（MFA） 48
对于需要长期凭证的用例，应在需要时更新访问密钥 48
遵循最佳实践以保护根用户凭证 49
应用最低权限权限 49
Amazon 托管策略及转向最低权限权限入门 50
使用 IAM Access Analyzer 根据访问活动生成最低权限策略 50
定期查看并移除未使用的用户、角色、权限、策略和凭证 50
使用 IAM policy 中的条件进一步限制访问权限 50
使用 IAM Access Analyzer 验证对资源的公共和跨账户访问 50
使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 51
跨多个账户建立权限护栏机制 51
使用权限边界在账户内委派权限管理 51
根用户最佳实践 51
保护您的根用户凭证，以防止未经授权的使用 53
使用强根用户密码来加强访问保护 53
使用多重身份验证（MFA）保护您的根用户登录安全 53
请勿为根用户创建访问密钥 54

尽可能使用多人审批进行根用户登录 54
将组电子邮件地址用于根用户凭证 54
限制对账户恢复机制的访问 54
保护您的 Amazon Organizations 账户的根用户凭证 55
监控访问权限和使用情况 56
商用案例 57
Example Corp 的初始设置 57
将 IAM 与 Amazon EC2 结合使用的使用案例 58
将 IAM 与 Amazon S3 结合使用的使用案例 59
教程 62
使用角色委派跨 Amazon Web Services 账户 的访问权限 62
注意事项 63
先决条件 64
在目标账户中创建角色 65
向角色授予访问权限 68
通过切换角色测试访问权限 70
其他资源 75
摘要 75
创建客户托管策略 75
先决条件 76
步骤 1：创建策略 76
步骤 2：附加策略 77
步骤 3：测试用户访问权限 77
相关资源 78
摘要 78
使用基于属性的访问控制 (ABAC) 78
教程概述 79
先决条件 80
步骤 1：创建测试用户 81
步骤 2：创建 ABAC 策略 83
步骤 3：创建角色 86
步骤 4：测试创建密钥 87
步骤 5：测试查看密钥 90
步骤 6：测试可扩展性 92
步骤 7：测试更新和删除密钥 93
摘要 95

相关资源 95
将 SAML 会话标签用于 ABAC 95
使 用户能够管理其凭证和 MFA 设置 99
先决条件 100
步骤 1：创建策略以强制 MFA 登录 101
步骤 2：将策略附加到您的测试用户组 102
步骤 3：测试您的用户的访问权限 102
相关资源 104
使用 Amazon CloudFormation 创建 SAML IdP 104
先决条件 105
创建 IdP 105
验证 IdP 108
删除资源 108
模板详细信息 108
模板 109
使用 Amazon CloudFormation 创建 SAML 联合角色 113
先决条件 113
创建联合角色 113
测试联合角色 116
删除资源 116
模板详细信息 117
模板 117
使用 Amazon CloudFormation 创建 SAML IdP 和联合角色 123
先决条件 124
创建 资源 124
测试联合身份验证 127
删除资源 128
模板详细信息 128
模板 128
身份 138
IAM 根用户 138
IAM 用户 138
IAM 用户组 139
IAM 角色 139
Amazon Web Services 账户根用户 139
集中管理成员账户的根访问权限 140

集中根访问 140
执行特权任务 144
根用户的 MFA 149
更改密码 156
重置丢失或遗忘的根用户密码 158
创建根用户的访问密钥 160
删除根用户的访问密钥 162
需要根用户凭证的任务 163
其他资源 165
Users 166
Amazon 如何标识 IAM 用户 166
IAM 用户和凭证 167
IAM 用户和权限 168
IAM 用户和账户 168
作为服务账户的 IAM 用户 168
IAM 用户如何登录 Amazon 169
创建用户 173
查看 IAM 用户 174
重命名用户 176
移除用户 178
控制用户对控制台的访问 182
更改用户权限 183
用户密码 190
管理 访问密钥 207
多重身份验证 233
服务特定凭证 306
查找未使用的凭证 315
生成凭证报告 318
CodeCommit 的 IAM 凭证 325
管理服务器证书 327
用户组 332
创建 IAM 组 333
查看 IAM 组 335
编辑 IAM 组中的用户 336
将策略附加到用户组 338
重命名用户组 338

删除 IAM 组 340
角色 341
何时创建 IAM 用户（而不是角色） 342
角色术语和概念 343
其他资源 346
混淆代理人问题 346
常见场景 356
角色创建 366
角色管理 411
担任角色的方法 442
身份提供程序和 Amazon 中的联合身份验证 597
使用 IAM Identity Center 的联合身份验证 599
使用 IAM 的联合身份验证 599
使用 Amazon Cognito 身份池的联合身份验证 600
其他资源 600
常见场景 600
OIDC 联合身份验证 605
SAML 2.0 联合身份验证 627
将 Amazon 身份联合到外部服务 663
常见使用案例 663
工作方式 664
开始使用出站身份联合验证 665
了解令牌声明 671
使用 IAM 策略控制访问权限 675
临时安全凭证 677
Amazon STS 和 Amazon 区域 677
临时凭证的常见情形 677
使用临时凭证的示例应用程序 679
有关临时凭证的其他资源 680
比较 Amazon STS 凭证 680
服务持有者令牌 683
请求临时安全凭证 684
将临时凭证用于 Amazon 资源 694
临时安全凭证的权限 697
管理 Amazon Web Services 区域中的 Amazon STS 731
Amazon STS 区域和端点 734

启用自定义身份凭证代理程序控制台访问 738
IAM 资源的标签 749
选择 Amazon 标签命名约定 750
在 IAM 和 Amazon STS 中进行标记的规则 751
标记 IAM 用户 753
标记 IAM 角色 756
标记客户管理型策略 759
标记 OIDC 身份提供者 762
标记 IAM SAML 身份提供者 764
标记实例配置文件 767
标记服务器证书 769
标记虚拟 MFA 设备 771
传递会话标签 773
访问权限管理 788
访问管理资源 790
策略和权限 790
策略类型 791
策略和根用户 796
JSON 策略概述 796
授予最低权限 801
托管策略与内联策略 802
数据边界 811
权限边界 816
身份与资源 829
使用策略控制访问 831
使用标签控制对用 IAM 用户和角色的访问 842
使用标签控制对 Amazon 资源的访问 844
跨账户资源访问 848
转发访问会话 854
策略示例 857
管理 IAM 策略 930
其他资源 931
创建 IAM policy 931
策略验证 940
策略测试 941
添加或删除身份权限 953

IAM 策略版本控制 964
编辑 IAM 策略 968
删除 IAM 策略 977
使用访问信息优化权限 983
策略摘要 1557
策略摘要（服务列表） 1558
策略摘要中的访问级别 1567
服务摘要（操作列表） 1570
操作摘要（资源列表） 1574
策略摘要示例 1578
访问 IAM 资源所需的权限 1587
用于管理 IAM 身份的权限 1587
在 Amazon Web Services 管理控制台中工作的权限 1589
向 Amazon 帐户授予权限 1589
某种服务访问其他服务的权限 1590
所需的操作 1590
IAM 的示例策略 1591
临时委派 1595
概述 1595
临时委派的工作原理 1595
发起临时委派请求 1597
审查临时委派请求 1599
撤销临时委派访问权限 1600
管理委派请求的权限 1601
通知 1602
CloudTrail 1603
合作伙伴的临时委派 1604
概述 1604
工作原理 1595
合作伙伴资格认证 1605
激活流程 1605
合作伙伴问卷 1606
了解您的集成 1607
了解权限 1610
策略评估指南 1612
策略模板 1615

构建您的集成 1625
代码示例 1633
IAM 1639
基本功能 1645
场景 2430
Amazon STS 2921
基本功能 2922
场景 2953
安全性 3204
Amazon 安全凭证 3205
安全性注意事项 3206
以编程方式访问 3207
Amazon 安全审核指南 3209
何时执行安全审查 3209
审核准则 3210
审核 Amazon 账户凭证 3210
审核 IAM 用户 3210
审核 IAM 组 3211
审核 IAM 角色 3211
查看您的 SAML 和 OpenID Connect（OIDC）的 IAM 提供商 3211
审核移动应用程序 3211
有关审核 IAM policy 的提示 3212
数据保护 3213
IAM 和 Amazon STS 中的数据加密 3214
IAM 和 Amazon STS 中的密钥管理 3214
IAM 和 Amazon STS 中的互联网络流量保密性 3215
日志记录和监控 3215
使用 CloudTrail 录入事件 3216
在 CloudTrail 中跟踪特权任务 3241
合规性验证 3244
故障恢复能力 3244
IAM 恢复能力的最佳实践 3246
基础结构安全性 3246
配置和漏洞分析 3247
Amazon 托管策略 3247
IAMReadOnlyAccess 3248

IAMUserChangePassword 3248
IAMAccessAnalyzerFullAccess 3248
IAMAccessAnalyzerReadOnlyAccess 3250
AccessAnalyzerServiceRolePolicy 3251
IAMAuditRootUserCredentials 3252
IAMCreateRootUserPassword 3253
IAMDeleteRootUserCredentials 3254
S3UnlockBucketPolicy 3256
SQSUnlockQueuePolicy 3257
策略更新 3257
IAM 外部的安全功能 3262
IAM Access Analyzer 3264
识别与外部实体共享的资源 3264
识别对关键业务资源的内部访问 3266
识别授予 IAM 用户和角色的未使用访问权限 3267
根据 Amazon 最佳实践验证策略 3267
根据指定的安全标准验证策略 3267
生成策略 3267
IAM Access Analyzer 定价 3268
调查发现 3268
调查发现的工作原理 3270
开始使用 IAM Access Analyzer 3273
调查发现控制面板 3288
查看调查发现 3292
筛选调查发现 3295
归档发现结果 3300
解决调查发现 3301
错误调查发现 3303
支持的资源类型 3308
委派管理员 3315
存档规则 3316
使用 EventBridge 监控 3318
Security Hub CSPM 集成 3329
使用 Cloudtrail 进行日志记录 3336
筛选条件键 3339
使用服务关联角色 3350

预览访问 3352
在 Amazon S3 控制台中预览访问权限 3352
使用 IAM Access Analyzer API 预览访问 3354
检查验证策略 3356
自定义策略检查的工作原理 3357
检查新访问的参考策略的示例 3358
检查失败的自定义策略检查 3358
使用基本策略检查进行验证 3358
策略检查参考 3361
使用自定义策略检查进行验证 3486
IAM Access Analyzer 策略生成 3487
策略生成的工作原理 3487
服务和操作级别的信息 3488
需知信息 3488
所需权限 3489
基于 CloudTrail 活动生成策略（控制台） 3492
使用其他账户中的 Amazon CloudTrail 数据生成策略 3495
基于 CloudTrail 活动生成策略（Amazon CLI） 3499
基于 CloudTrail 活动生成策略（Amazon API） 3499
IAM Access Analyzer 策略生成服务 3500
IAM Access Analyzer 配额 4049
排查 IAM 问题 4052
我无法登录我的 Amazon 账户 4052
我丢失了访问密钥 4052
策略变量不起作用 4053
我所做的更改可能不会立即可见 4053
我没有权限执行：iam:DeleteVirtualMFADevice 4054
如何安全地创建 IAM 用户？ 4055
其他资源 4055
拒绝访问错误消息 4055
当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝” 4056
当我使用临时安全凭证发送请求时，收到了“访问被拒绝” 4057
拒绝访问示例 4058
根用户问题 4065
IAM 策略 4066
使用可视化编辑器排除故障 4067

排查策略摘要问题 4071
策略管理故障排查 4079
JSON 策略文档故障排查 4080
通行密钥和 FIDO 安全密钥 4086
我无法启用我的 FIDO 安全密钥 4086
我无法使用 FIDO 安全密钥登录 4087
我的 FIDO 安全密钥已丢失或损坏 4087
其它问题 4088
IAM 角色 4088
我无法代入角色 4088
我的 Amazon 账户中出现新角色 4090
我无法编辑或删除我的 Amazon Web Services 账户 中的角色 4091
我无权执行：iam:PassRole 4091
为什么我无法在 12 小时会话中担任角色？ (Amazon CLI、Amazon API) 4091
当我尝试在 IAM 控制台中切换角色时收到错误 4092
我的角色具有一个允许我执行操作的策略，但出现“访问被拒绝”错误 4092
服务未创建角色的默认策略版本 4092
控制台中没有服务角色的使用案例 4094
IAM 和 Amazon EC2 4095
当尝试启动实例时，我没有在 Amazon EC2 控制台的 IAM 角色列表中看到该角色 4095
我的实例的凭证适用于错误的角色 4096
当时尝试调用 AddRoleToInstanceProfile 时，发生 AccessDenied 错误 4096
Amazon EC2：在我尝试使用角色启动实例时，出现 AccessDenied 错误 4096
我无法访问我的 EC2 实例的临时安全凭证 4097
IAM 树形子目录中 info 文档的错误意味着什么？ 4098
IAM 和 Amazon S3 4099
如何授权匿名访问 Amazon S3 存储桶？ 4099
我以 Amazon Web Services 账户 根用户身份登录。为什么我无法访问自己账户下的 Amazon S3 存储桶？ 4099
SAML 2.0 联合身份验证 4099
无效的 SAML 响应 4100
需要 RoleSessionName 4101
没有 AssumeRoleWithSAML 授权 4101
无效的 RoleSessionName 字符 4102
无效的源身份字符 4102
无效的响应签名 4102

私钥无效 4102
无法删除私有密钥 4103
无法删除私有密钥，因为密钥 ID 与私有密钥不匹配 4103
无法担任角色 4103
无法解析元数据 4103
无法更新身份提供商 4104
无法将断言加密模式设置为“必需”，因为未提供私有密钥 4104
无法在同一请求中添加和删除私钥 4104
指定的提供商不存在 4104
DurationSeconds 超过 MaxSessionDuration 4104
已达到私有密钥限制 2 4105
响应不包含所需受众 4105
IAM 如何与其他 Amazon 服务协同工作 4106
使用 Amazon CloudFormation 创建 IAM 资源 4106
IAM 和 Amazon CloudFormation 模板 4107
了解有关 Amazon CloudFormation 的更多信息 4107
将 Amazon CloudShell 与 IAM 配合使用 4107
获取 Amazon CloudShell 的 IAM 权限 4108
与 IAM 交互 4108
使用 Amazon SDK 4110
参考 4111
使用 Amazon 资源名称（ARN）标识 Amazon 资源 4111
ARN 格式 4111
查找资源的 ARN 格式 4112
ARN 中的路径 4113
IAM 标识符 4114
易记名称和路径 4115
IAM ARN 4115
唯一标识符 4122
了解唯一 ID 前缀 4123
获取唯一标识符 4124
IAM 和 Amazon STS 配额 4125
IAM 名称要求 4125
IAM 对象配额 4126
IAM Access Analyzer 配额 4127
IAM Roles Anywhere 限额 4127

STS 请求配额 4128
IAM 和 STS 字符限制 4129
双堆栈端点支持 4132
IAM 双堆栈端点支持 4133
Amazon STS 双堆栈端点支持 4133
接口 VPC 端点 4134
VPC 端点可用性 4135
为 IAM 创建 VPC 端点 4136
为 Amazon STS 创建 VPC 端点 4137
使用 IAM 的服务 4137
使用 IAM 的服务 4139
更多信息 4177
Amazon 签名版本 4 4181
Amazon SigV4 工作原理 4182
Amazon SigV4a 的工作原理 4182
何时签署请求 4183
为什么签署请求 4183
其他资源 4183
SigV4 请求元素 4184
身份验证方法 4186
创建已签名的请求 4192
请求签名示例 4204
排查 SigV4 问题 4206
策略参考 4210
JSON 元素参考 4211
策略评估逻辑 4312
策略语法 4336
工作职能的 Amazon 托管策略 4344
全局条件键 4357
IAM 条件键 4430
操作、资源和条件键 4476
资源 4477
身份 4477
证书（密码、访问密钥和 MFA 设备） 4477
权限与策略 4478
联合和委托 4478

IAM 和其他 Amazon 产品 4478
将 IAM 与 Amazon EC2 结合使用 4478
将 IAM 与 Amazon S3 结合使用 4479
将 IAM 与 Amazon RDS 结合使用 4479
将 IAM 与 Amazon DynamoDB 结合使用 4479
一般安全实践 4479
一般 资源 4480
发出 HTTP 查询请求 4481
端点 4481
必须使用 HTTPS 4482
签署 IAM API 请求 4482
文档历史记录 4483

什么是 IAM？

Follow us on Twitter

Amazon Identity and Access Management (IAM) 是一种 Web 服务，可以帮助您安全地控制对 Amazon 资源的访问。借助 IAM，您可以管理控制用户可访问哪些 Amazon 资源的权限。可以使用 IAM 来控制谁通过了身份验证（准许登录）并获得授权（具有相应权限）来使用资源。IAM 提供了控制您 Amazon Web Services 账户 身份验证和授权所需的基础设施。

身份

当您创建 Amazon Web Services 账户 时，最初使用的是一个对所有 Amazon Web Services 服务和资源拥有完全访问权限的登录身份（称为 Amazon Web Services 账户根用户）。我们强烈建议不要使用根用户进行日常任务。有关要求根用户凭证的任务，请参阅《IAM 用户指南》中的需要根用户凭证的任务。

除了根用户之外，使用 IAM 还可以设置其他身份，例如管理员、分析师和开发人员，并且可以授予其访问成功完成其任务所需资源的访问权限。

访问管理

在 IAM 中设置用户后，他们将使用其登录凭证向 Amazon 进行身份验证。通过匹配登录凭证与受 Amazon Web Services 账户 信任的主体（IAM 用户、Amazon STS 联合主体、IAM 角色或应用程序）来进行身份验证。接下来，请求授予主体对资源的访问权限。如果用户已被授予资源的相应资源，则根据授权请求授予访问权限。例如，当您首次登录控制台并进入控制台主页时，您并未访问特定服务。当您选择一项服务时，授权请求将发送至该服务，并查看您的身份是否在授权用户列表中，正在执行哪些策略来控制授予的访问级别，以及任何其他可能生效的策略。授权请求可以由您 Amazon Web Services 账户 内的主体提出，也可以由您信任的其他 Amazon Web Services 账户 提出。

获得授权后，主体可以对您 Amazon Web Services 账户 里的资源采取行动或执行操作。例如，主体可以启动新的 Amazon Elastic Compute Cloud 实例、修改 IAM 群组成员资格或删除 Amazon Simple Storage Service 存储桶。

> **Tip**
>
> Amazon 培训和认证提供了介绍 IAM 的 10 分钟视频：
> Amazon Identity and Access Management 简介。

服务可用性

IAM 和很多其他 Amazon 服务一样，具备最终一致性。IAM 通过复制 Amazon 在全球的数据中心内多个服务器上的数据实现高可用性。如果成功请求更改某些数据，则更改会提交并安全存储。不过，更改必须跨 IAM 复制，这需要时间。此类更改包括创建或更新用户、组、角色或策略。在应用程序的关键、高可用性代码路径中，我们不建议进行此类 IAM 更改。而应在不常运行的、单独的初始化或设置例程中进行 IAM 更改。另外，在生产工作流程依赖这些更改之前，请务必验证更改已传播。有关更多信息，请参阅 我所做的更改可能不会立即可见。

服务费用信息

Amazon Identity and Access Management（IAM）、Amazon IAM Identity Center 和 Amazon Security Token Service（Amazon STS）是为您的 Amazon 账户提供的功能，不会另外收费。只有在您使用 IAM 用户或 Amazon STS 临时安全证书访问其他 Amazon 服务时才会向您收取费用。

IAM Access Analyzer 外部访问分析没有额外费用。但是，您需要为未使用的访问分析和客户策略检查支付费用。有关 IAM Access Analyzer 的收费和价格的完整列表，请参阅 IAM Access Analyzer 定价。

有关其他 Amazon 产品的定价信息，请参阅 Amazon Web Services 定价页面。

与其他 Amazon 服务集成

IAM 已与很多 Amazon 服务集成。有关与 IAM 配合使用的 Amazon 服务列表以及这些服务支持的 IAM 功能，请参阅 使用 IAM 的 Amazon 服务。

为什么应该使用 IAM？

Amazon Identity and Access Management 是一款用于安全管理 Amazon 资源访问权限的强大工具。使用 IAM 的主要优点之一是能够向您的 Amazon 账户授予共享访问权限。此外，IAM 还允许您分配精细的权限，从而可让您精确控制不同用户可以对特定资源执行的操作。这种访问控制级别对于维护 Amazon 环境的安全至关重要。IAM 还具有其他几项安全功能。您可以添加多重身份验证（MFA）以获得额外的保护层，以及利用身份联合验证无缝集成来自公司网络或其他身份提供商的用户。IAM 还与 Amazon CloudTrail 集成，并且提供详细的日志记录和身份信息，以支持审计和合规性要求。通过利用这些功能，您可以协助确保对关键 Amazon 资源的访问受到严格控制和安全保障。

对您 Amazon Web Services 账户 的共享访问权限

您可以向其他人员授予管理和使用您 Amazon 账户中的资源的权限，而不必共享您的密码或访问密钥。

精细权限

您可以针对不同资源向不同人员授予不同权限。例如，您可以允许某些用户完全访问 Amazon Elastic Compute Cloud (Amazon EC2)、Amazon Simple Storage Service (Amazon S3)、Amazon DynamoDB、Amazon Redshift 以及其他 Amazon 服务。对于另一些用户，您可以允许仅针对某些 Amazon S3 存储桶的只读访问权限，或是仅管理某些 Amazon EC2 实例的权限，或是访问您的账单信息但无法访问任何其他内容的权限。

在 Amazon EC2 上运行的应用程序针对 Amazon 资源的安全访问权限

您可以使用 IAM 功能安全地为 EC2 实例上运行的应用程序提供凭证。这些凭证为您的应用程序提供权限以访问其他 Amazon 资源。示例包括 S3 存储桶和 DynamoDB 表。

多重身份验证 (MFA)

您可以向您的账户和各个用户添加双重身份验证以实现更高安全性。借助 MFA，您或您的用户不仅必须提供使用账户所需的密码或访问密钥，还必须提供来自经过特殊配置的设备的代码。如果您已将 FIDO 安全密钥与其他服务配合使用，则该密钥应该具有 Amazon 支持的配置，您可以使用 WebAuthn 实现 MFA 安全性。有关更多信息，请参阅使用密钥或安全密钥的受支持配置

联合身份

您可以允许已在其他位置（例如，在您的企业网络中或通过互联网身份提供商）获得密码的用户获取对您 Amazon Web Services 账户 的访问权限。这些用户被授予符合 IAM 最佳实践建议的临时凭证。使用联合身份验证增强 Amazon 账户的安全性。

实现保证的身份信息

如果您使用 Amazon CloudTrail，则会收到日志记录，其中包括有关对您账户中的资源进行请求的人员的信息。这些信息基于 IAM 身份。

PCI DSS 合规性

IAM 支持由商家或服务提供商处理、存储和传输信用卡数据，而且已经验证符合支付卡行业 (PCI) 数据安全标准 (DSS)。有关 PCI DSS 的更多信息，包括如何请求 Amazon PCI Compliance Package 的副本，请参阅 PCI DSS 第 1 级。

何时使用 IAM？

Amazon Identity and Access Management 是一项核心基础设施服务，为 Amazon 中基于身份的访问控制提供基础。每次访问 Amazon 账户时都使用 IAM。您使用 IAM 的方式将取决于组织内的具体职责和工作职能。Amazon 服务的用户利用 IAM 访问其日常工作所需的 Amazon 资源，管理员则授予相应的权限。另一方面，IAM 管理员负责管理 IAM 身份并编写策略来控制对资源的访问。无论您的角色如何，每次对 Amazon 资源进行身份验证和授权访问时，您都会与 IAM 进行交互。这可能涉及以 IAM 用户身份登录、担任 IAM 角色或利用联合身份验证实现无缝访问。了解各种 IAM 功能和使用案例是有效管理 Amazon 环境安全访问的关键举措。在创建策略和权限时，IAM 提供一种灵活而精细的方法。除了基于身份的策略（指定用户或角色可以访问的操作和资源）之外，您还可以定义信任策略来控制哪些主体可以担任角色。通过配置这些 IAM 策略，您可以帮助确保用户和应用程序具有执行其所需任务的适当权限级别。

当您执行不同的工作职能时

Amazon Identity and Access Management 是一项核心基础设施服务，为 Amazon 中基于身份的访问控制提供基础。每次访问 Amazon 账户时都使用 IAM。

使用 IAM 的方式因您可以在 Amazon 中执行的操作而异。

- 服务用户 – 如果使用 Amazon 服务来完成工作，则您的管理员会为您提供所需的凭证和权限。当您使用更多高级功能来完成工作时，您可能需要额外权限。了解如何管理访问权限有助于您向管理员请求适合的权限。
- 服务管理员 – 如果您在公司负责管理 Amazon 资源，则您可能具有对 IAM 的完全访问权限。您有责任确定您的服务用户应访问哪些 IAM 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。
- IAM 管理员 – 如果您是 IAM 管理员，您可以管理 IAM 身份并编写策略以管理对 IAM 的访问。

当您获得授权访问 Amazon 资源时

身份验证是您使用身份凭证登录 Amazon 的方法。您必须作为 Amazon Web Services 账户根用户、IAM 用户或通过担任 IAM 角色进行身份验证。

对于编程访问，Amazon 提供了 SDK 和 CLI 来对请求进行加密签名。有关更多信息，请参阅《IAM 用户指南》中的适用于 API 请求的 Amazon 签名版本 4。

当您以 IAM 用户身份登录时

IAM 用户是对某个人员或应用程序具有特定权限的一个身份。建议使用临时凭证，而非具有长期凭证的 IAM 用户。有关更多信息，请参阅《IAM 用户指南》中的要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon。

IAM 组指定一组 IAM 用户，便于更轻松地对大量用户进行权限管理。有关更多信息，请参阅《IAM 用户指南》中的 IAM 用户使用案例。

当您代入 IAM 角色时

IAM 角色是具有特定权限的身份，可提供临时凭证。您可以通过从用户切换到 IAM 角色（控制台）或调用 Amazon CLI 或 Amazon API 操作来担任角色。有关更多信息，请参阅《IAM 用户指南》中的担任角色的方法。

IAM 角色对于联合用户访问、临时 IAM 用户权限、跨账户访问、跨服务访问以及在 Amazon EC2 上运行的应用程序非常有用。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

当您创建策略和权限时

您可通过创建策略 (这是列出用户可执行的操作以及操作可以影响的资源的文档) 向用户授予权限。默认情况下会拒绝未显式允许的任何操作或资源。可将策略创建并附加到主体（用户、用户组、用户代入的角色以及资源）。

您可以将这些策略与 IAM 角色一起使用：

- 信任策略：定义哪些主体能够在何种条件下担任该角色。信任策略是适用于 IAM 角色的一种特定类型的基于资源的策略。角色只能具有一个信任策略。
- 基于身份的策略（内联和托管）– 这些策略定义了角色的用户能够对哪些资源执行（或拒绝执行）的权限。

使用 IAM 基于身份的策略示例 帮助您的 IAM 身份定义权限。在找到所需的策略后，请选择查看该策略以查看该策略的 JSON 版本。您可以将该 JSON 策略文档用作自己策略的模板。

> **Note**
>
> 如果您使用 IAM Identity Center 管理用户，则可以在 IAM Identity Center 中分配权限集，而非将权限策略附加到主体。当您向组或 Amazon IAM Identity Center 中的用户 分配权限集

时，IAM Identity Center 会在每个账户中创建相应的 IAM 角色，并将权限集中指定的策略附加到这些角色中。IAM Identity Center 管理该角色，并允许您定义的授权用户代入该角色。如果您修改权限集，IAM Identity Center 会确保相应的 IAM policy 和角色也相应更新。
有关 IAM Identity Center 的更多信息，请参阅 Amazon IAM Identity Center 用户指南中的什么是 IAM Identity Center？。

我如何管理 IAM？

在 Amazon 环境中管理 Amazon Identity and Access Management 涉及利用各种工具和界面。最常用的方法是通过 Amazon Web Services 管理控制台，这一基于 Web 的界面可让您执行各种 IAM 管理任务，从创建用户和角色到配置权限。

对于更熟悉命令行界面的用户，Amazon 提供了两组命令行工具 - Amazon Command Line Interface 和 Amazon Tools for Windows PowerShell。这些工具可让您直接从终端发出与 IAM 相关的命令，通常比导航控制台更高效。此外，Amazon CloudShell 允许您使用与控制台登录相关联的权限，直接从 Web 浏览器运行 CLI 或 SDK 命令。

除了控制台和命令行之外，Amazon 还提供适用于各种编程语言的软件开发工具包（SDK），使您能够将 IAM 管理功能直接集成到自己的应用程序中。或者，您可以使用 IAM Query API（可让您直接向服务发布 HTTPS 请求）以编程方式访问 IAM。利用这些不同的管理方法，您可以灵活地将 IAM 整合到现有的工作流程和过程中。

使用 Amazon Web Services 管理控制台

Amazon 管理控制台是一款 Web 应用程序，其中包含并引用了多种用于管理 Amazon 资源的服务控制台。首次登录时，您会看到控制台主页。通过控制台主页可以访问各个服务控制台，此外还可以一站式访问执行 Amazon 相关任务所需的信息。登录控制台后可以使用哪些服务和应用程序，取决于您有权访问哪些 Amazon 资源。您可以通过代入角色、成为已获得相关权限的组成员或获得显式授权来获得资源权限。对于独立 Amazon 账户，根用户或 IAM 管理员负责配置对资源的访问权限。对于 Amazon Organizations，管理账户或委托管理员负责配置对资源的访问权限。

如果您计划让人使用 Amazon 管理控制台来管理 Amazon 资源，我们建议您为用户配置临时凭证，以遵循 最佳安全实践。代入角色的 IAM 用户、联合主体和 IAM Identity Center 中的用户拥有临时凭证，而 IAM 用户和根用户拥有长期凭证。根用户凭证提供对 Amazon Web Services 账户的完全访问权限，而其他用户拥有的凭证提供对 IAM policy 授权的资源的访问权限。

不同类型的 Amazon Web Services 管理控制台 用户会有不同的登录体验。

- IAM 用户和根用户通过 Amazon 主登录 URL（https://signin.aws.amazon.com）登录。登录之后，这些用户可以访问账户中已获得权限的资源。

  您必须使用根用户电子邮件地址和密码才能以根用户身份登录。

  您必须拥有 Amazon Web Services 账户号码或别名、IAM 用户名和 IAM 用户密码才能以 IAM 用户身份登录。

  我们建议您仅在需要长期凭证的特定情况（例如用于紧急访问）下使用账户中的 IAM 用户，并且仅将根用户用于 需要根用户凭证的任务。

  为方便起见，Amazon 登录页面使用浏览器 Cookie 记住 IAM 用户名和账户信息。下次用户转到 Amazon Web Services 管理控制台 中的任何页面时，控制台会使用 cookie 将用户重定向到账户登录页面。

  完成会话后应退出控制台，以防止之前的登录信息被重复使用。
- IAM Identity Center 用户使用其组织特有的特定 Amazon Web Services 访问门户登录。登录后，这些用户可以选择要访问的账户或应用程序。如果选择访问某个账户，则可以选择要在管理会话中使用的权限集。
- 在与 Amazon Web Services 账户关联的外部身份提供者中管理的 OIDC 和 SAML 联合主体使用自定义企业访问门户登录。用户可用的 Amazon 资源取决于其组织选择的策略。

> **Note**
>
> 可以要求根用户、IAM 用户和 IAM Identity Center 中的用户通过 Amazon 的多重身份验证（MFA）验证身份后才能获得对 Amazon 资源的访问权限，从而增强安全性。启用 MFA 后，您还必须有权访问 MFA 设备才能登录。

要详细了解不同用户如何登录管理控制台，请参阅《Amazon 登录用户指南》中的 Sign in to the Amazon Management Console。

Amazon 命令行工具

您可以使用 Amazon 命令行工具，在系统的命令行中发出命令以执行 IAM 和 Amazon 任务。与控制台相比，使用命令行更快、更方便。如果要构建执行 Amazon 任务的脚本，命令行工具也会十分有用。

Amazon 提供两组命令行工具：Amazon Command Line Interface（Amazon CLI）和 Amazon Tools for Windows PowerShell。有关安装和使用 Amazon CLI 的更多信息，请参阅 Amazon Command Line

Interface 用户指南。有关安装和使用 Tools for Windows PowerShell 的信息，请参阅 Amazon Tools for PowerShell 用户指南。

登录控制台后，您可以从浏览器使用 Amazon CloudShell 来运行 CLI 或 SDK 命令。访问 Amazon 资源的权限取决于您登录控制台时使用的凭证。根据您的经验，您可能会发现使用 CLI 来管理 Amazon Web Services 账户 更高效。有关更多信息，请参阅 将 Amazon CloudShell 与 Amazon Identity and Access Management 结合使用。

Amazon 命令行界面（CLI）和软件开发工具包（SDK）

IAM Identity Center 和 IAM 用户在通过 CLI 或相关 SDK 中的应用程序编程接口（API）进行身份验证时，将使用不同的方法来验证其凭证。

凭证和配置设置位于不同位置（例如，系统或用户环境变量、本地 Amazon 配置文件）或在命令行上显式声明为参数。某些位置优先于其他位置。

IAM Identity Center 和 IAM 都提供了可用于 CLI 或 SDK 的访问密钥。IAM Identity Center 访问密钥是可以自动刷新的临时凭证，建议优先使用这种访问密钥，而不是与 IAM 用户关联的长期访问密钥。

要使用 CLI 或 SDK 来管理 Amazon Web Services 账户，您可以在浏览器中使用 Amazon CloudShell。如果您使用 CloudShell 来运行 CLI 或 SDK 命令，则必须首先登录控制台。访问 Amazon 资源的权限取决于您登录控制台时使用的凭证。根据您的经验，您可能会发现使用 CLI 来管理 Amazon Web Services 账户更高效。

对于应用程序开发，您可以将 CLI 或 SDK 下载到您的计算机上，然后从命令提示符或 Docker 窗口登录。在这种情况下，您可以在 CLI 脚本或 SDK 应用程序中配置身份验证和访问凭证。您可以通过不同的方式配置对资源的编程访问权限，具体取决于环境和可用的访问权限。

- 使用 Amazon 服务进行本地代码身份验证时，推荐的方法是使用 IAM Identity Center 和 IAM Roles Anywhere
- 对 Amazon 环境中运行的代码，推荐的身份验证方法是使用 IAM 角色或使用 IAM Identity Center 凭证。

使用 Amazon Web Services 访问门户登录时，您可以从选择权限集的起始页获取短期凭证。这些凭证有明确的有效期，不会自动刷新。如果要使用此类凭证，请在登录 Amazon 门户后，选择 Amazon Web Services 账户，然后选择该权限集。选择命令行或编程访问权限以查看可用于以编程方式或通过 CLI 访问 Amazon 资源的选项。有关这些方法的更多信息，请参阅《IAM Identity Center 用户指南》的 Getting and refreshing temporary credentials。这些凭证通常在应用程序开发期间用于快速测试代码。

我们建议在自动化访问您的 Amazon 资源时使用会自动刷新的 IAM Identity Center 凭证。如果您已在 IAM Identity Center 中配置了用户和权限集，则可以使用 `aws configure sso` 命令来使用命令行向导，这将有助于识别可供您使用的凭证并将其存储在配置文件中。有关配置文件配置的更多信息，请参阅《Amazon 命令行界面版本 2 用户指南》中的 使用 aws configure sso 向导配置您的配置文件。

> **Note**
>
> 许多示例应用程序使用与 IAM 用户或根用户关联的长期访问密钥。在试用学习时，您只应在沙盒环境中使用长期凭证。查看 长期访问密钥的替代方案，并计划尽快将代码过渡到使用替代凭证，例如 IAM Identity Center 凭证或 IAM 角色。转换代码后，请删除访问密钥。

要了解有关配置 CLI 的更多信息，请参阅《Amazon 命令行界面版本 2 用户指南》中的 安装或更新最新版本的 Amazon CLI，以及《Amazon 命令行界面用户指南》中的 身份验证和访问凭证

要了解有关配置 SDK 的更多信息，请参阅《Amazon SDK 和工具参考指南》中的 IAM Identity Center authentication，以及《Amazon SDK 和工具参考指南》中的 IAM Roles Anywhere。

使用 Amazon SDK

Amazon 提供的 SDK（软件开发工具包）包含各种编程语言和平台（Java、Python、Ruby、.NET、iOS、Android 等）的库和示例代码。开发工具包提供了可通过编程方式访问 IAM 和 Amazon 的便捷方式。例如，软件开发工具包执行以下类似任务：加密签署请求、管理错误以及自动重试请求。有关 Amazon 软件开发工具包的信息（包括如何下载及安装），请参阅适用于 Amazon Web Services 的工具页面。

使用 IAM Query API

您可以使用 IAM Query API（可让您直接向服务发布 HTTPS 请求）以编程方式访问 IAM 和 Amazon。当您使用 Query API 时，必须添加代码，才能使用您的凭证对请求进行数字化签名。有关更多信息，请参阅 使用 HTTP 查询请求调用 IAM API 和 IAM API 参考。

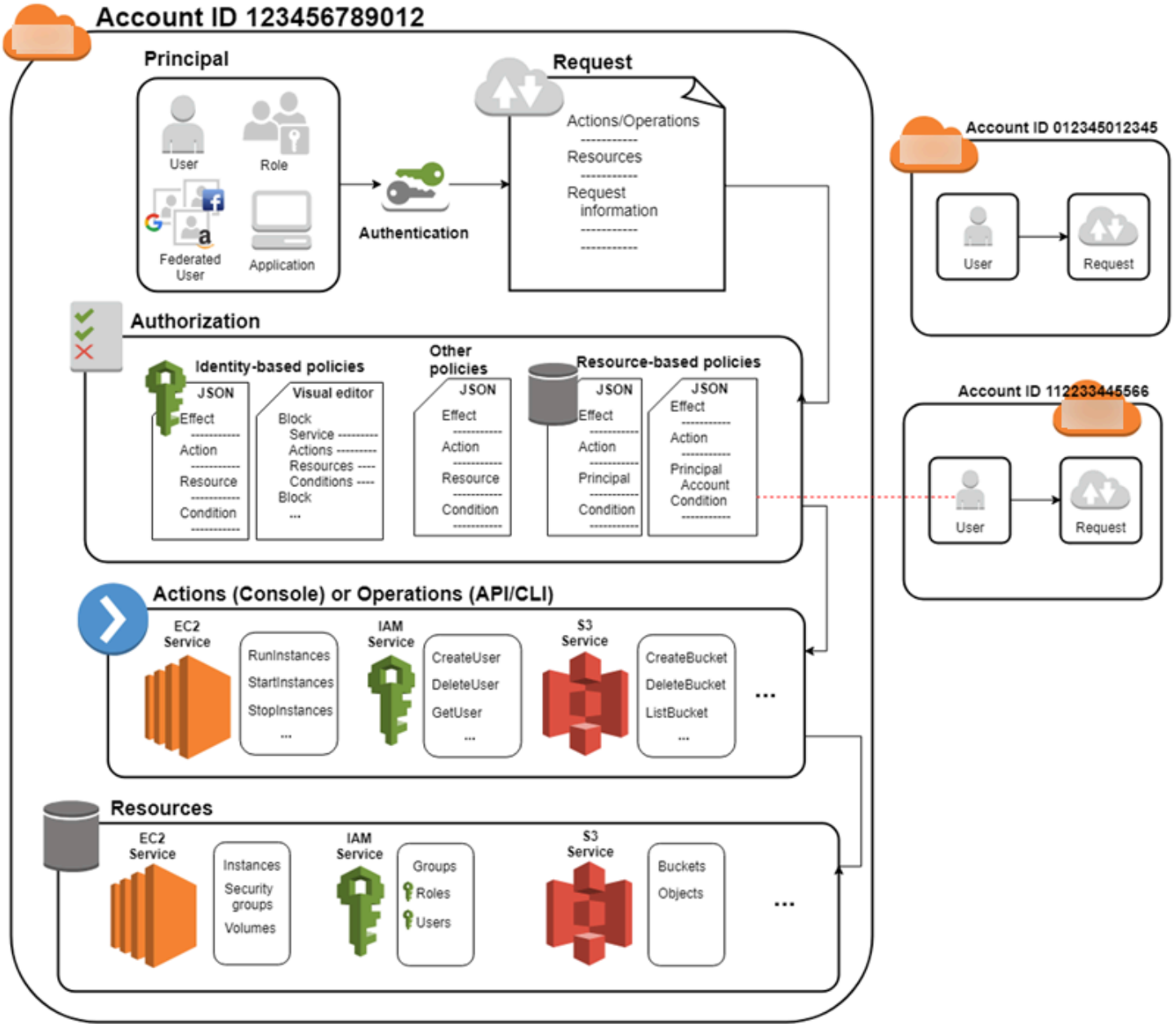
IAM 的工作原理

Amazon Identity and Access Management 提供了控制您 Amazon Web Services 账户 身份验证和授权所需的基础设施。

首先，人类用户或应用程序使用其登录凭证与 Amazon 进行身份验证。IAM 将登录凭证与受 Amazon Web Services 账户 信任的主体（IAM 用户、Amazon STS 联合用户主体、IAM 角色或应用程序）进行匹配，验证访问 Amazon 的主体。

接下来，IAM 请求授予主体对资源的访问权限。IAM 根据授权请求授予或拒绝访问。例如，当您首次登录控制台并进入控制台主页时，您并未访问特定服务。选择一项服务后，将向 IAM 发送该服务的授权请求。IAM 将验证您的身份是否在授权用户列表中，确定哪些策略控制授予的访问权限级别，并评估任何其他可能生效的策略。您的 Amazon Web Services 账户 或您信任的其他 Amazon Web Services 账户 内的主体可以提出授权请求。

获得授权后，主体可以对您 Amazon Web Services 账户 中的资源执行操作。例如，主体可以启动新的 Amazon Elastic Compute Cloud 实例、修改 IAM 群组成员资格或删除 Amazon Simple Storage Service 存储桶。下图通过 IAM 基础设施说明了这一过程：



请求的组成部分

在主体尝试使用 Amazon Web Services 管理控制台、Amazon API 或 Amazon CLI 时，该主体将向 Amazon 发送请求。请求包含以下信息：

- 操作：主体想要执行的操作，例如 Amazon Web Services 管理控制台 中的操作，或者 Amazon CLI 或 Amazon API 中的操作。
- 资源：主体请求对其执行操作的 Amazon 资源对象。
- 主体 – 已使用实体（用户或角色）发送请求的人员或应用程序。有关主体的信息包括权限策略。

- 环境数据：有关 IP 地址、用户代理、SSL 启用状态和时间戳的信息。
- 资源数据：与请求的资源相关的数据，例如 DynamoDB 表名或 Amazon EC2 实例上的标签。

Amazon 将请求信息收集到请求上下文中，IAM 会对其进行评估以授权该请求。

如何对主体进行身份验证

主体使用其凭证登录 Amazon，IAM 会对其进行身份验证，以允许主体向 Amazon 发送请求。某些服务（如 Amazon S3 和 Amazon STS）允许来自匿名用户的特定请求。不过，它们是该规则的例外情况。每种类型的用户都要经过身份验证。

- 根用户：用于身份验证的登录凭证是您用于创建 Amazon Web Services 账户 的电子邮件地址和当时指定的密码。
- 联合主体：您的身份提供者会对您进行身份验证并将您的凭证传递给 Amazon，您无需直接登录 Amazon。IAM Identity Center 和 IAM 都支持身份联合验证。
- Amazon IAM Identity Center 目录 中的用户（非联合用户）：直接在 IAM Identity Center 默认目录中创建的用户使用 Amazon Web Services 访问门户登录并提供您的用户名和密码。
- IAM 用户：您可以通过提供账户 ID 或别名、用户名和密码进行登录。要对 API 或 Amazon CLI 中的工作负载进行身份验证，您可以通过承担角色使用临时凭证，也可以通过提供访问密钥和私密访问密钥来使用长期凭证。

  要了解有关 IAM 实体的更多信息，请参阅 IAM 用户 和 IAM 角色。

Amazon 建议您对所有用户使用多重身份验证（MFA）以提高您账户的安全性。要了解有关 MFA 的更多信息，请参阅 IAM 中的 Amazon 多重身份验证。

授权和权限策略基础知识

授权是指主体拥有完成其请求所需的权限。在授权期间，IAM 使用请求上下文中的值来识别适用于该请求的策略。然后，它使用策略来确定是允许还是拒绝请求。IAM 将大多数权限策略存储为指定主体实体权限的 JSON 文档。

有多种类型的策略可以影响授权请求。要向用户提供访问您账户中 Amazon 资源的权限，您可以使用基于身份的策略。基于资源的策略可以授予跨账户访问权限。要在另一个账户中发出请求，此其他账户中的策略必须允许访问资源，并且您用于发出请求的 IAM 实体必须具有允许该请求基于身份的策略。

IAM 检查应用于请求上下文的每个策略。IAM 策略评估使用显式拒绝，这意味着如果一个权限策略包含拒绝操作，则 IAM 将拒绝整个请求并停止评估。由于请求默认被拒绝，因此适用的权限策略必须允许请求的每个部分，IAM 才能授权您的请求。单个账户中对于请求的评估逻辑遵循以下基本规则：

- 默认情况下，所有请求都将被拒绝。（通常，始终允许使用 Amazon Web Services 账户根用户 凭证创建的访问该账户资源的请求。）
- 任何权限策略（基于身份或基于资源）中的显式允许将覆盖此默认值。
- Amazon Organizations 服务控制策略（SCP）或资源控制策略（RCP）、IAM 权限边界或会话策略的存在将覆盖允许。如果存在其中一个或多个策略类型，它们必须都允许请求。否则，将隐式拒绝它。有关 SCP 和 RCP 更多信息，请参阅《Amazon Organizations 用户指南》中的 Authorization policies in Amazon Organizations。
- 任何策略中的显式拒绝都会覆盖所有策略中的允许。

要了解更多信息，请参阅策略评估逻辑。

在 IAM 对主体进行身份验证和授权后，IAM 将通过评估适用于主体的权限策略来批准其请求中的操作。每项 Amazon 服务都定义了其支持的操作，包括可以对资源执行的操作，例如，查看、创建、编辑和删除该资源。适用于主体的权限策略必须包含执行操作必需的操作。要了解有关 IAM 如何评估权限策略的更多信息，请参阅 the section called “策略评估逻辑”。

服务定义了主体可以对每个资源执行的一组操作。创建权限策略时，请确保包含您希望用户能够执行的操作。例如，IAM 支持针对用户资源的 40 多种操作，包括以下基本操作：

- `CreateUser`
- `DeleteUser`
- `GetUser`
- `UpdateUser`

此外，您还可以在权限策略中指定条件，当请求满足指定条件时，允许访问资源。例如，您可能希望策略语句在特定日期之后生效，或者在 API 请求中存在特定值时控制访问。要指定条件，您可以使用策略语句的 `Condition` 元素。

在 IAM 批准请求中的操作后，主体便可使用您账户中的相关资源。资源是位于服务中的对象。示例包括 Amazon EC2 实例、IAM 用户和 Amazon S3 存储桶。如果主体创建请求以对未包含在权限策略中的资源执行操作，则服务会拒绝该请求。例如，如果您有权删除 IAM 角色，但请求删除 IAM 组；您无权删除 IAM 组，因此请求失败。要了解不同 Amazon 服务支持哪些操作、资源和条件键，请参阅 Amazon 服务的操作、资源和条件键。

比较 IAM 身份和凭证

Amazon Identity and Access Management 中管理的身份是 IAM 用户、IAM 角色和 IAM 组。这些是 Amazon 与 Amazon Web Services 账户 一起创建的根用户之外的身份。

强烈建议您不使用根用户执行日常任务，即使是管理任务。相反，可以预置其他用户并授予他们执行必要任务所需的权限。可以通过如下方式添加用户：将人员添加到 IAM Identity Center 目录，将外部身份提供商与 IAM Identity Center 或 IAM 联合，或者创建最低权限 IAM 用户。

> **Important**
>
> 在北京和宁夏区域，没有根用户的概念。所有用户都是 IAM 用户，包括创建 Amazon 账户的用户。

为了获得额外的安全性，我们建议集中根访问，以帮助您集中保护使用 Amazon Organizations 管理的 Amazon Web Services 账户 的根用户凭证。集中管理成员账户的根访问权限 允许您集中移除和防止长期的根用户凭证恢复，从而防止大规模的意外根访问。启用集中根访问后，您可以假设一个特权会话，以对成员账户执行操作。

设置用户之后，您可以授予特定人员对 Amazon Web Services 账户 的访问权限，并为他们提供访问资源的权限。

作为最佳实践，Amazon 建议您要求人类用户担任 IAM 角色访问 Amazon，从而他们可使用临时凭证。如果您在 IAM Identity Center 目录中管理身份，或者使用与身份提供商的联合身份验证，则应遵循最佳实践。

术语

通常在使用 IAM 身份时使用这些术语：

IAM 资源

IAM 服务存储以下资源。您可以在控制台中添加、编辑和移除它们。

- IAM 用户
- IAM 组
- IAM 角色
- 权限策略

- 身份提供者对象

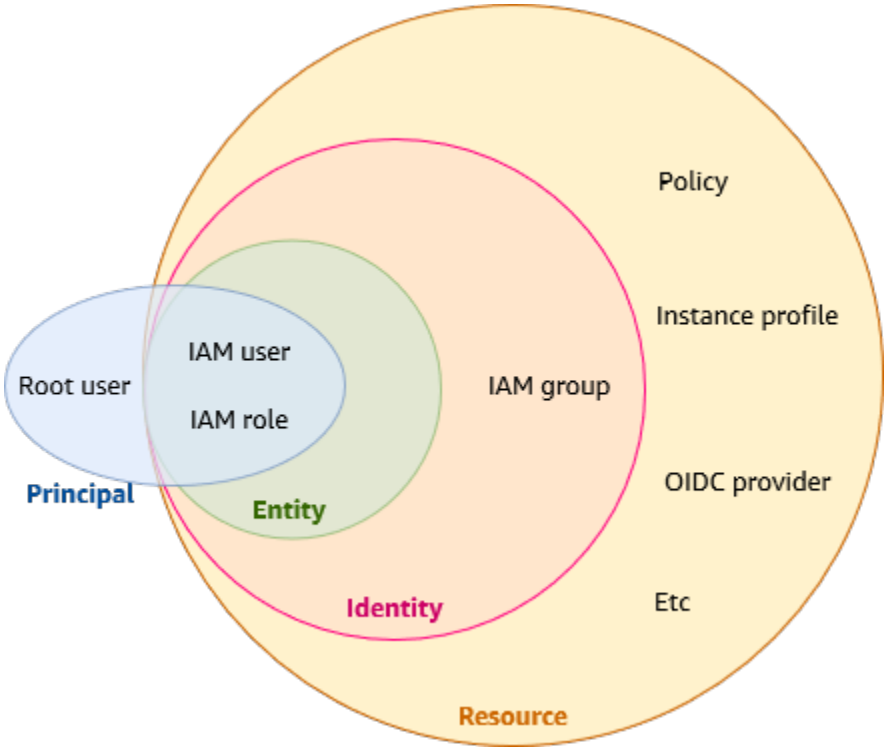
IAM 实体

Amazon 用于身份验证的 IAM 资源。在基于资源的策略中指定该实体作为主体。

- IAM 用户
- IAM 角色

IAM 身份

在策略中授权执行操作和访问资源的 IAM 资源。身份包括 IAM 用户、IAM 组和 IAM 角色。



主体

可对 Amazon 资源发出操作请求的 Amazon Web Services 账户根用户、IAM 用户或 IAM 角色。主体包括人类用户、工作负载、联合主体和担任的角色。在身份验证后，IAM 会根据主体类型授予主体永久或临时凭证，以向 Amazon 发出请求。

人类用户也称为人类身份，例如应用程序的人员、管理员、开发人员、操作员和使用者。

工作负载是可提供商业价值的一系列资源和代码，例如应用程序、过程、操作工具和其他组件。

联合主体是指其身份和凭证由其他身份提供者（例如 Active Directory、Okta 或 Microsoft Entra）管理的用户。

IAM 角色是可在账户中创建的 IAM 身份，该身份具有特定权限，可确定该身份可执行和不可执行的操作。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。

IAM 可向 IAM 用户和根用户授予长期凭证和 IAM 角色临时凭证。Amazon IAM Identity Center 中的用户、OIDC 和 SAML 联合主体登录 Amazon 后将担任 IAM 角色，获得其授予的临时凭证。作为最佳实践，我们建议您要求人类用户和工作负载使用临时凭证访问 Amazon 资源。

IAM 用户和 IAM Identity Center 中用户之间的区别

IAM 用户不是单独的账户；他们是您账户中的个人用户。每个用户都可以有自己的密码可用于访问 Amazon Web Services 管理控制台。您还可以为每个用户创建单独的访问密钥，以便用户可以发出编程请求以使用账户中的资源。

IAM 用户及其访问密钥将拥有 Amazon 资源的长期凭证。IAM 用户的主要用途是使无法使用 IAM 角色的工作负载能够使用 API 或 CLI 向 Amazon 服务发出编程请求。

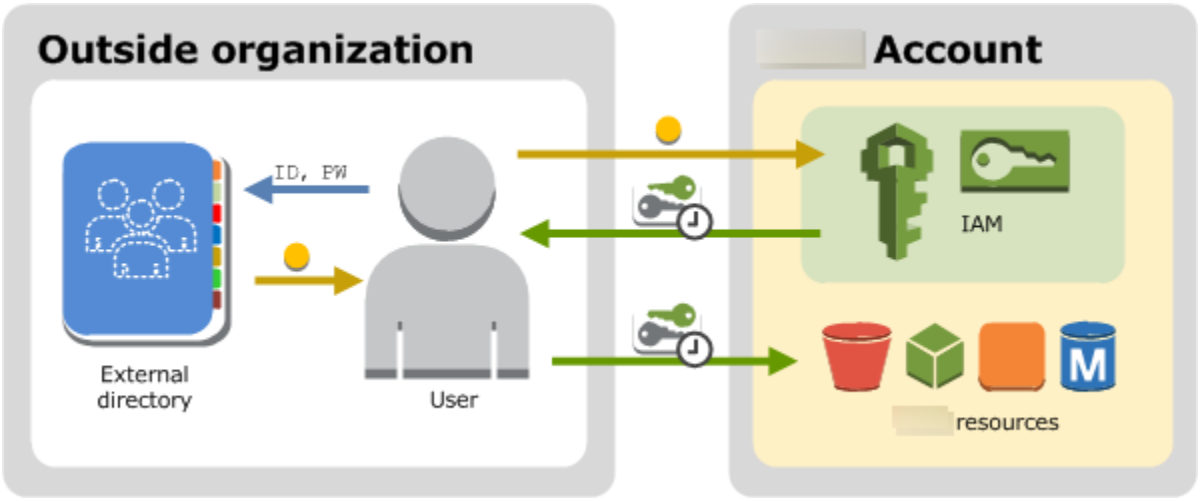
> **Note**
>
> 对于需要具有编程访问权限和长期凭证的 IAM 用户的场景，我们建议在需要时更新访问密钥。有关更多信息，请参阅 更新访问密钥。

员工身份（人员）是 Amazon IAM Identity Center 中的用户，其根据所执行的角色具有不同的权限需求，并且可以在组织内的各种 Amazon Web Services 账户 中工作。如果您的用例需要访问密钥，则可以使用 Amazon IAM Identity Center 中的用户 支持这些应用场景。通过 Amazon Web Services 访问门户登录的人员可以使用 Amazon 资源的短期凭证获取访问密钥。对于集中式访问权限管理，我们建议使用 Amazon IAM Identity Center（IAM Identity Center）来管理对您账户的访问权限以及这些账户中的其他权限。IAM Identity Center 会自动配置 Identity Center 目录作为您的默认身份源，您可以在其中添加人员和组，并分配其对您 Amazon 资源的访问级别。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 Amazon IAM Identity Center。

这两种类型用户之间的主要区别在于，IAM Identity Center 中的用户在登录 Amazon 时会自动担任 IAM 角色，然后才能访问管理控制台或 Amazon 资源。每次用户登录 Amazon 时，IAM 角色都会授予临时凭证。对于使用 IAM 角色登录的 IAM 用户，其必须具有担任和切换角色的权限，并且必须在访问 Amazon 账户后显式选择切换到其想要担任的角色。

现有身份源的联合用户

如果您组织中的用户在登录到公司网络时已通过身份验证，则不必为其创建单独的 IAM 用户或 IAM Identity Center 中的用户。相反，您可以使用 IAM 或 Amazon IAM Identity Center 将这些用户身份联合到 Amazon 中。OIDC 和 SAML 联合主体将担任 IAM 角色，该角色授予其访问特定资源的权限。有关角色的更多信息，请参阅 角色术语和概念。



联合身份验证在以下情况尤其有用：

- 您的用户已存在于公司目录中。

  如果您的公司目录与安全断言标记语言 2.0 (SAML 2.0) 兼容，则可以配置公司目录以便为用户提供对 Amazon Web Services 管理控制台的单一登录 (SSO) 访问。有关更多信息，请参阅 临时凭证的常见情形。

  如果您的公司目录与 SAML 2.0 不兼容，则可以创建身份凭证代理程序应用程序以便为用户提供对 Amazon Web Services 管理控制台 的单点登录（SSO）访问。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。

  如果您的公司目录是 Microsoft Active Directory，则可以使用 Amazon IAM Identity Center 连接 Active Directory 中的自我管理目录或 Amazon Directory Service 中的目录，以便在公司目录与您 Amazon Web Services 账户 之间建立信任。

  如果您使用外部身份提供者（IdP）（例如 Okta 或 Microsoft Entra）管理用户，则可以使用 Amazon IAM Identity Center 在 IdP 和 Amazon Web Services 账户 之间建立信任。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的连接到外部身份提供者。

- 您的用户已有 Internet 身份。

如果您创建的移动应用程序或基于 Web 的应用程序可以允许用户通过 Internet 身份提供商 (如 Login with Amazon、Facebook、Google 或任何与 OpenID Connect (OIDC) 兼容的身份提供商) 标识自己，则应用程序可以使用联合访问 Amazon。有关更多信息，请参阅 OIDC 联合身份验证。

> **Tip**
>
> 要使用与 Internet 身份提供商的联合身份，我们建议使用 Amazon Cognito。

提供用户访问权限的不同方法

以下是可以提供您的 Amazon 资源访问权限的几种方式。

用户访问类型	何时使用？	哪里有更多信息？
使用 IAM Identity Center 为人员（例如您的员工用户）提供 Amazon 资源的单点登录访问权限	IAM Identity Center 提供了一个集中位置，将用户的管理及其对 Amazon Web Services 账户 和云应用程序的访问汇集在一起。 您可以在 IAM Identity Center 中设置身份存储，也可以与现有身份提供者（IdP）一起配置联合身份验证。安全最佳实践建议向人类用户授予 Amazon 资源的有限凭证。 人员可以获得更简单的登录体验，并且您可以控制其通过单个系统访问资源。IAM Identity Center 支持多重身份验证（MFA），以提高账户的安全性。	有关设置 IAM Identity Center 的更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的入门。 有关在 IAM Identity Center 中使用 MFA 的更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的多重身份验证。

用户访问类型	何时使用？	哪里有更多信息？
人类用户（例如您的员工用户）使用 IAM 身份提供商（IdP）访问 Amazon 服务的联合访问权限	IAM 支持与 OpenID Connect (OIDC) 或者 SAML 2.0 (Security Assertion Markup Language 2.0) 兼容的 IdPs。创建 IAM 身份提供者后，必须创建一个或多个可以动态分配给联合主体的 IAM 角色。	有关 IAM 身份提供程序和联合身份验证的更多信息，请参阅 身份提供程序和 Amazon 中的联合身份验证。
Amazon Web Services 账户间的跨账户访问权限	您想要与其他 Amazon Web Services 账户 中的用户共享对特定 Amazon 资源的访问权限。 角色是授予跨账户访问权限的主要方式。但是，某些 Amazon 服务支持基于资源的策略，允许您将策略直接附加到资源（而不是使用角色作为代理）。	有关 IAM 角色的更多信息，请参阅 IAM 角色。 有关服务关联角色的更多信息，请参阅创建服务相关角色。 有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务。查找在服务相关角色列中具有是值的服务。要查看该服务的服务相关角色文档，请选择该列中与是有关的链接。

用户访问类型	何时使用？	哪里有更多信息？
您的 Amazon Web Services 账户 中指定 IAM 用户的长期凭证	您可能有一些特定的使用案例需要带有 Amazon 中 IAM 用户的长期凭证。您可以使用 IAM 在您的 Amazon Web Services 账户 中创建这些 IAM 用户，并使用 IAM 管理他们的权限。部分使用场景包括： • 无法使用 IAM 角色的工作负载 • 需要通过访问密钥进行编程访问的第三方 Amazon 客户端 • Amazon CodeCommit 或 Amazon Keyspaces 的服务特定凭证 • Amazon IAM Identity Center 不适用于您的账户，而且您没有其他身份提供商 根据 最佳实践，对于需要具有 编程访问权限和长期凭证 的 IAM 用户的场景，我们建议在需要时更新访问密钥。有关更多信息，请参阅 更新访问密钥。	有关设置 IAM 用户的信息，请参阅 在 Amazon Web Services 账户 中创建 IAM 用户。 有关 IAM 用户访问密钥的更多信息，请参阅 管理 IAM 用户的访问密钥。。 有关 Amazon CodeCommit 或 Amazon Keyspaces 的服务特定凭证的更多信息，请参阅 CodeCommit 的 IAM 凭证：Git 凭证、SSH 密钥和 Amazon 访问密钥 和 将 IAM 与 Amazon Keyspaces（Apache Cassandra 兼容）结合使用。

支持编程用户访问

如果用户需要在 Amazon Web Services 管理控制台之外与 Amazon 交互，则需要编程式访问权限。授予编程式访问权限的方法取决于访问 Amazon 的用户类型：

- 如果您在 IAM Identity Center 中管理身份，则 Amazon API 需要一个配置文件，而 Amazon Command Line Interface 需要一个配置文件或环境变量。

- 如果您有 IAM 用户，则 Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	Option	更多信息
员工身份 （在 IAM Identity Center 中管理的人员和用户）	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	对于 Amazon CLI，请按照《Amazon IAM Identity Center 用户指南》中获取用于 CLI 访问的 IAM 角色凭证中的说明进行操作。 对于 Amazon API，请按照《Amazon SDK 和工具参考指南》中 SSO 凭证中的说明进行操作。
IAM 用户	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照将临时凭证用于 Amazon 资源中的说明进行操作。
IAM 用户	使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。 （不推荐使用）	按照管理 IAM 用户的访问密钥中的说明进行操作。
联合身份验证主体	使用 Amazon STS API 操作创建具有临时安全凭证（包括访问密钥对和会话令牌）的新会话。	有关 API 操作的说明，请参阅 the section called “请求临时安全凭证”

权限和策略如何提供访问管理

Amazon Identity and Access Management（IAM）的访问管理部分帮助定义主体实体可在账户内执行的操作。主体实体是指使用 IAM 实体（IAM 用户或 IAM 角色）进行身份验证的人员或应用程序。访问管理通常称为授权。您在 Amazon 中通过创建策略并将其附加到 IAM 身份（IAM 用户、IAM 组或 IAM 角色）或 Amazon 资源来管理访问权限。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在主体使用 IAM 实体（IAM 用户或 IAM 角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略在 Amazon 中存储为 JSON 文档。有关策略类型和用法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

策略和账户

如果您管理 Amazon 中的单个账户，则使用策略定义该账户中的权限。如果您管理跨多个账户的权限，则管理 IAM 用户的权限会比较困难。您可以将 IAM 角色、基于资源的策略或访问控制列表 (ACL) 用于跨账户权限。但是，如果您拥有多个账户，那我们建议您改用该 Amazon Organizations 服务来帮助您管理这些权限。有关更多信息，请参阅《Amazon Organizations 用户指南》中的什么是 Amazon Organizations？。

策略和用户

IAM 用户是Amazon Web Services 账户中的身份。当您创建 IAM 用户时，他们无法访问您账户中的任何内容，直到您向他们授予权限。向 IAM 用户授予权限的方法是创建基于身份的策略，这是附加到 IAM 用户或 IAM 用户所属 IAM 组的策略。下面的示例演示一个 JSON 策略，该策略允许 IAM 用户对 `us-east-2` 区域内 `123456789012` 账户中的 `Books` 表执行所有 Amazon DynamoDB 操作（`dynamodb:*`）。

JSON

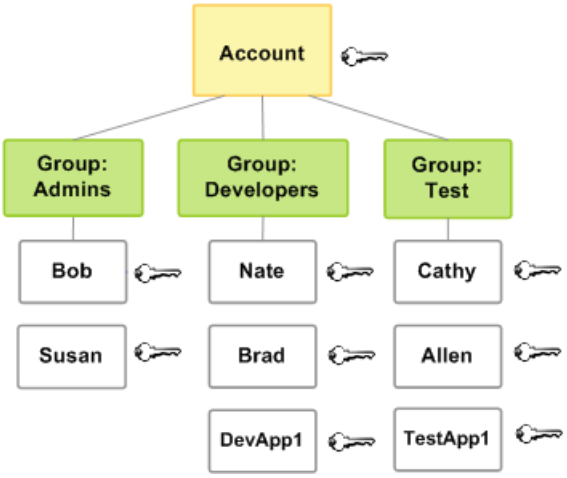
```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "dynamodb:*",
    "Resource": "arn:aws:dynamodb:us-east-2:123456789012:table/Books"
  }
}
```

在将此策略附加到您的 IAM 用户后，该用户有权在 DynamoDB 实例的 Books 表中执行所有操作。大多数 IAM 用户都有多个策略，这些策略组合在一起代表其授予的总权限。

默认情况下会拒绝策略未显式允许的操作或资源。例如，如果上述策略是附加到用户的单个策略，则该用户可以对 Books 表执行 DynamoDB 操作，但不能对其他表执行这些操作。同样，不允许用户在 Amazon EC2、Amazon S3 或任何其他 Amazon 服务中执行任何操作，因为策略中未包含使用这些服务的权限。

策略和 IAM 组

可以将 IAM 用户组织为 IAM 组，然后将策略附加到该 IAM 组。这种情况下，各 IAM 用户仍有自己的凭证，但是 IAM 组中的所有 IAM 用户都具有附加到该 IAM 组的权限。使用 IAM 组可更轻松地管理权限。



IAM 用户或 IAM 组可以附加授予不同权限的多个策略。在这种情况下，策略的组合决定了主体的有效权限。如果主体对操作和资源没有显式 Allow 权限，则主体没有这些权限。

联合用户会话和角色

联合主体无法通过与 IAM 用户相同的方式在您的 Amazon Web Services 账户中获得永久身份。要向联合主体分配权限，可以创建称为角色的实体，并为角色定义权限。当 SAML 或 OIDC 联合主体登录 Amazon 时，该用户会与角色关联，被授予角色中定义的权限。有关更多信息，请参阅 为第三方身份提供者创建角色。

基于身份和基于资源的策略

基于身份的策略是附加到 IAM 身份（如 IAM 用户、组或角色）的权限策略。基于资源的策略是附加到资源（如 Amazon S3 存储桶或 IAM 角色信任策略）的权限策略。

基于身份的策略控制身份可以在哪些条件下对哪些资源执行哪些操作。基于身份的策略可以进一步分类：

- 托管策略 – 基于身份的独立策略，可附加到您的 Amazon Web Services 账户 中的多个用户、组和角色。您可以使用两个类型的托管策略：
  - Amazon 托管策略——由 Amazon 创建和管理的托管策略。如果您刚开始使用策略，建议先使用 Amazon 托管策略。
  - 客户管理型策略 – 您在 Amazon Web Services 账户 中创建和管理的管理型策略。与 Amazon 托管策略相比，客户托管策略可以更精确地控制策略。您可以在可视化编辑器中创建、编辑和验证 ，它是一项 IAM policy，也可以直接创建 JSON 策略文档以创建和编辑该策略。有关更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限 和 编辑 IAM 策略。
- 内联策略——由您创建和管理的策略，直接嵌入在单个用户、组或角色中。大多数情况下，我们不建议使用内联策略。

基于资源的策略控制指定的主体可以在何种条件下对该资源执行哪些操作。基于资源的策略是内联策略，没有基于资源的托管策略。要启用跨账户存取，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。

IAM 服务支持一种基于资源的策略（称为角色信任策略），您可将这种策略附加到 IAM 角色。由于 IAM 角色同时是支持基于资源的策略的身份和资源，因此，您必须同时将信任策略和基于身份的策略附加到 IAM 角色。信任策略定义哪些主体实体（账户、用户、角色和 Amazon STS 联合用户主体）可以代入该角色。要了解 IAM 角色如何与其他基于资源的策略不同，请参阅 IAM 中的跨账户资源访问。

要了解哪些服务支持基于资源的策略，请参阅使用 IAM 的 Amazon 服务。要了解基于资源的策略的更多信息，请参阅 基于身份的策略和基于资源的策略。

使用 ABAC 授权根据属性定义权限

基于属性的访问权限控制（ABAC）是一种授权策略，该策略基于属性来定义权限。Amazon 将这些属性称为标签。您可以将标签附加到 IAM 资源（包括 IAM 实体（IAM 用户和 IAM 角色））以及 Amazon 资源。您可以为 IAM 委托人创建单个 ABAC 策略或者一小组策略。您可以将这些 ABAC 策略设计为在主体的标签与资源标签匹配时允许操作。ABAC 的属性系统既提供高级用户上下文，又提供精细的访问控制。由于 ABAC 是基于属性的，因此它可以对数据或应用程序执行动态授权，从而实时授予或撤销访问权限。ABAC 在扩展环境中以及身份或资源策略管理变得复杂的情况下非常有用。

例如，您可以创建具有 `access-project` 标签键的三个 IAM 角色。将第一个 IAM 角色的标签值设置为 `Heart`，第二个为 `Star`，第三个为 `Lightning`。然后，您可以使用单个策略，在 IAM 角色和

Amazon 资源具有标签值 access-project 时允许访问。有关演示如何在 Amazon 中使用 ABAC 的详细教程，请参阅IAM 教程：根据标签定义访问 Amazon 资源的权限。要了解支持 ABAC 的服务，请参阅 使用 IAM 的 Amazon 服务。
下图说明应用于主体的标签必须与应用于资源的标签相匹配，才能向用户授予该资源的权限。标签可应用于 IAM 组、资源组、个人用户和单个资源。

ABAC 与传统 RBAC 模型的比较

IAM 中使用的传统授权模型是基于角色的访问控制（RBAC）。RBAC 根据用户的工作职能（或称角色）定义权限，这与 IAM 角色不同。IAM 确实包含任务函数的管式策略，能够将权限分配给 RBAC 模型中的任务函数。

在 IAM 中，您通过为不同工作职能创建不同策略来实施 RBAC。然后，您可将策略附加到身份（IAM 用户、IAM 组或 IAM 角色）。作为最佳实践，您向工作职能授予所需的最小权限。这会导致最低权限访问。每个工作职能策略都列出了分配给该策略的身份可以访问的特定资源。使用传统 RBAC 模型的缺点是，当您或您的用户向您的环境添加新资源时，您必须更新策略以允许访问这些资源。

例如，假设您的员工在处理三个项目，名为 Heart、Star 和 Lightning。您可以为每个项目创建一个 IAM 角色。然后，您将策略附加到各个 IAM 角色，定义允许担任该 IAM 角色的任何用户可以访问的资源。如果员工更换了公司中的工作，您可向其分配不同的 IAM 角色。您可以将用户或计划分配给多个 IAM 角色。但是，Star 项目可能需要额外的资源，例如新的 Amazon EC2 容器。在这种情况下，您必须更新附加到 Star IAM 角色的策略，来指定新的容器资源。否则，系统将不允许 Star 项目成员访问新的容器。
下图说明基于角色的访问控制要求为每个身份分配一个基于工作职能的特定策略来访问不同的资源。

相比传统 RBAC 模型，ABAC 具备以下优势：

- ABAC 权限随着创新扩展。它不再需要管理员更新现有策略以允许对新资源的访问。例如，假设您使用 access-project 标签指定了 ABAC 策略。开发人员使用 access-project = Heart 标签的 IAM 角色。当 Heart 项目中的员工需要额外的 Amazon EC2 资源时，开发人员可以使用 access-project = Heart 标签创建新 Amazon EC2 实例。这样，Heart 项目中的任何员工可以启动和停止这些实例，因为其标签值匹配。
- ABAC 需要较少的策略。由于您无需为不同工作职能创建不同策略，需要创建的策略数量减少。这些策略更易于管理。
- 使用 ABAC，团队可以动态响应变化和增长。由于新资源的权限是根据属性自动授予的，因此您无需为身份手动分配策略。例如，如果您的公司已经使用 ABAC 支持 Heart 和 Star 项目，则可以轻松地添加新 Lightning 项目。IAM 管理员创建具有 access-project = Lightning 标签的新 IAM 角色。无需更改策略以支持新项目。有权担任该 IAM 角色的任何用户可以创建和查看使用 access-

`project = Lightning` 标记的实例。另一种情况是团队成员从 `Heart` 项目转到 `Lightning` 项目。为了向团队成员提供 `Lightning` 项目的访问权限，IAM 管理员将其分配给不同的 IAM 角色。无需更改权限策略。

- 使用 ABAC 可以实现精细权限。在您创建策略时，最佳实践是授予最小权限。使用传统 RBAC 时，您可编写允许访问特定资源的策略。但使用 ABAC 时，如果资源的标签与主体的标签匹配，则可以允许对所有资源执行操作。
- 通过 ABAC 使用来自您公司目录的员工属性。您可以配置 SAML 或 OIDC 提供商，以将会话标签传递给 IAM。当您的员工希望在 Amazon 中进行联合身份验证时，IAM 会将其属性应用于其最终的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。

有关演示如何在 Amazon 中使用 ABAC 的详细教程，请参阅IAM 教程：根据标签定义访问 Amazon 资源的权限。

IAM 入门

Amazon Identity and Access Management (IAM) 可以帮助您安全地控制对 Amazon Web Services (Amazon) 和您的账户资源的访问。IAM 也可以使您的登录凭证保持私密。您不用专门注册使用 IAM。使用 IAM 不会产生任何费用。

使用 IAM 为用户和角色等身份提供访问帐户中资源的权限。例如，您可以对您在 Amazon 外部管理的企业目录中的现有用户使用 IAM，或者您可以使用 Amazon IAM Identity Center 在 Amazon 中创建用户。联合身份代入已定义的 IAM 角色来访问所需的资源。有关 IAM Identity Center 的更多信息，请参阅 Amazon IAM Identity Center 用户指南中的什么是 IAM Identity Center？。

> **Note**
>
> IAM 已与多种 Amazon 产品集成。有关支持 IAM 的服务的列表，请参阅 使用 IAM 的 Amazon 服务。

要了解如何开始使用 Amazon、创建管理用户和 Amazon Organizations 以及使用多种服务来解决构建和启动您的第一个项目等问题，请参阅入门资源中心。

保护 IAM 用户

注册 Amazon Web Services 账户 后，启用多重身份验证（MFA）保护您的管理用户。有关说明，请参阅 IAM 用户指南中的 为 IAM 用户启用虚拟 MFA 设备（控制台）。

要授予其他用户访问您的 Amazon Web Services 账户 资源的权限，请创建 IAM 用户。为了保护 IAM 用户，请启用 MFA，并仅向 IAM 用户授予执行其任务必需的权限。

有关创建和保护 IAM 用户的更多信息，请参阅《IAM 用户指南》中的以下主题：

- 在您的 Amazon Web Services 账户中创建 IAM 用户
- 管理对 Amazon 资源的访问
- IAM 基于身份的策略示例

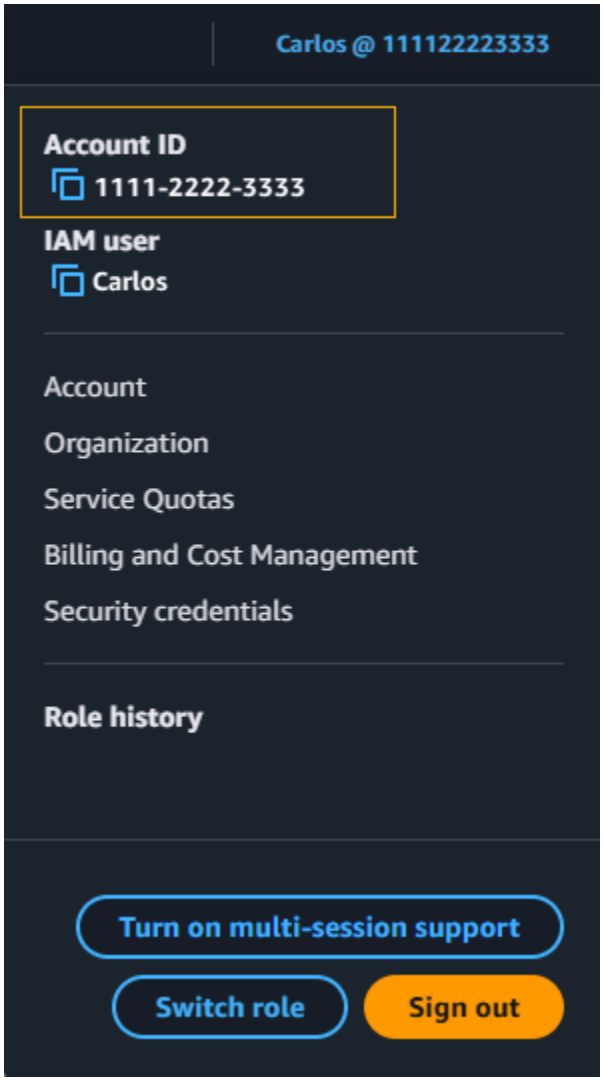
查看您的 Amazon Web Services 账户 ID

如果您已登录控制台，则可以使用以下方法查看您的 Amazon Web Services 账户 的账户 ID。

查看您的 Amazon Web Services 账户 ID

Console

当您转到 IAM 控制面板的 Amazon Web Services 账户 部分时，将显示 Amazon 账户 ID。您也可以在右上角的导航栏中查看您的账户 ID。选择您的用户名，账户 ID 会显示在您的用户名上方。



Amazon CLI

使用以下命令查看您的用户 ID、账户 ID 和用户 ARN：

- aws sts get-caller-identity

API

使用以下 API 查看您的用户 ID、账户 ID 和用户 ARN：

- GetCallerIdentity

为您的 Amazon Web Services 账户 ID 使用别名

账户 ID 是 12 位数字，用于唯一标识您的账户。账户中的 IAM 用户默认使用包含账户 ID 的 Web URL 登录。如果没有 URL，他们登录时可以在 Amazon 登录页面上提供账户 ID。

您的登录页面 URL 地址默认格式如下：

```
https://Your_Account_ID.signin.aws.amazon.com/console/
```

许多人发现单词比数字更容易记住，因此为您的账户 ID 创建别名可以帮助 IAM 用户更轻松地登录。

如果为您的 Amazon Web Services 账户 ID 创建一个 Amazon Web Services 账户 别名，您的登录页面 URL 地址格式类似如下示例。

```
https://Your_Account_Alias.signin.aws.amazon.com/console/
```

创建账户别名前的注意事项

- 您的 Amazon Web Services 账户 只能有一个别名。如果为您的 Amazon 账户创建新的别名，新别名将覆盖原有别名，包含原有别名的 URL 将失效。
- 账户别名必须仅包含数字、小写字母和连字符。有关 Amazon 账户实体的限制条件的更多信息，请参阅 IAM 和 Amazon STS 配额。
- 账户别名必须在给定网络分区内的所有 Amazon Web Services 产品中是唯一的名称。

  分区 是一组 Amazon 区域。每个 Amazon 账户的作用域为一个分区。

  以下是支持的分区：

  - `aws` - Amazon 区域
  - `aws-cn` – 中国区域
  - `aws-us-gov` - Amazon GovCloud (US) 区域

> **Note**
>
> 账户别名不是密码，它们将显示在面向公众的登录页面 URL 中。请勿在账户别名中包含任何敏感信息。

创建 Amazon Web Services 账户 别名后，包含 Amazon Web Services 账户 ID 的原始 URL 地址依然有效，可以使用。

创建账户别名

要执行下列步骤，您必须至少具有以下 IAM 权限：

- `iam:ListAccountAliases`
- `iam:CreateAccountAlias`

创建 Amazon Web Services 账户 别名

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择控制面板。
3. 在 Amazon 账户部分，找到账户别名，然后选择创建。如果别名已存在，则选择 Edit（编辑）。
4. 在对话框中，输入要用于别名的名称，然后选择保存更改。

Amazon CLI

运行以下命令：

- `aws iam create-account-alias`

API

要为 Amazon Web Services 管理控制台 登录页面 URL 创建别名，请调用以下操作：

- `CreateAccountAlias`

删除账户别名

要执行下列步骤，您必须至少具有以下 IAM 权限：

- iam:ListAccountAliases
- iam:DeleteAccountAlias

删除账户别名

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择控制面板。
3. 在Amazon账户部分，选择账户别名旁边的删除。

Amazon CLI

要删除 Amazon Web Services 账户 ID 别名，请运行以下命令：

- aws iam delete-account-alias

要确认账户别名已删除，请尝试通过运行以下命令显示您的 Amazon Web Services 账户 ID 别名：

- aws iam list-account-aliases

API

要删除 Amazon Web Services 账户 ID 别名，请调用以下操作：

- DeleteAccountAlias

要确认账户别名已删除，请尝试通过调用以下操作显示您的 Amazon Web Services 账户 ID 别名：

- ListAccountAliases

> **Note**
>
> 删除账户别名后，您账户的唯一登录 URL 基于您的账户 ID。任何连接到别名 URL 的尝试都将失败并且不会重定向。

规划对 Amazon 账户的访问权限

设置 Amazon 时，请规划希望人们如何访问您的 Amazon 账户和资源，以设置精心设计且安全的身份管理解决方案。

身份来源

根据 IAM 最佳实践，人类用户和工作负载在访问您的 Amazon 资源时应使用临时凭证。临时凭证将授予使用 IAM 角色访问资源的身份。联合到 IAM 的用户和 IAM Identity Center 内的用户（联合身份或在 IAM Identity Center 目录中创建）都使用 IAM 角色来访问资源。

在开始使用 Amazon 之前，请通过以下方式规划如何设置您的身份：

- 启用具有 Amazon Organizations 的 IAM Identity Center，并将 IAM Identity Center 中的用户直接添加到组织目录中。

  要了解如何将用户直接添加到 IAM Identity Center 组织目录，请参阅添加用户
- 将您现有的外部身份提供商与 IAM Identity Center 或 IAM 联合起来。

  要了解如何将外部身份提供商联合到 IAM Identity Center 组织目录，请使用相应的入门教程。

访问管理

确定您的用户将访问的 Amazon 资源和服务，并定义每个用户、组或角色所需的访问权限和策略。

- 如果您使用 IAM Identity Center，则会在组织的每个 Amazon 账户中自动创建 IAM 身份提供商以及 IAM 角色和权限策略。这些角色和权限与您在向特定应用程序或 Amazon 账户分配人员或组时指定的权限一致。

  有关更多信息，请参阅分配用户访问权限和设置对应用程序的单点登录访问权限。
- 如果您在 Amazon Web Services 账户 中将身份提供商直接与 IAM 联合，则必须创建一个供用户担任的角色和两条策略：一条信任策略，指定可以担任该角色的人员；一条权限策略，指定允许或拒绝担任该角色的人员访问的 Amazon 操作和资源。

  有关更多信息，请参阅 身份提供程序和 Amazon 中的联合身份验证。

IAM 用户的使用案例

您在 Amazon Web Services 账户 中创建的 IAM 用户拥有您可以直接管理的长期凭证。

在 Amazon 中管理访问权限时，IAM 用户通常并非最佳选择。在大多数使用案例中，您应该避免依赖 IAM 用户，原因有几个。

首先，IAM 用户是为个人账户设计的，因此随着组织的发展，他们无法很好地扩展。管理大量 IAM 用户的权限和安全很快就会成为一项挑战。

IAM 用户还缺乏其他 Amazon 身份管理解决方案所提供的集中可见性和审计功能。这可能会使维护安全性和监管合规性变得更具挑战性。

最后，使用更具可扩展性的身份管理方法，可以更轻松地实施安全最佳实践，例如多重身份验证、密码策略和角色分离。

建议不要依赖 IAM 用户，而是使用更强大的解决方案，例如具有 Amazon Organizations 的 IAM Identity Center 或来自外部提供商的联合身份。随着 Amazon 环境的增长，这些选项将为您提供更好的控制、安全性和运营效率。

因此，建议您仅对联合用户不支持的使用案例使用 IAM 用户。

以下列表确定 Amazon 中需要带 IAM 用户的长期凭证的特定使用案例。您可以使用 IAM 在您 Amazon 账户的伞形结构下创建这些 IAM 用户，并使用 IAM 管理他们的权限。

- 紧急访问您的 Amazon 账户
- 无法使用 IAM 角色的工作负载
  - Amazon CodeCommit访问
  - Amazon Keyspaces（Apache Cassandra 兼容）访问
- 第三方 Amazon 客户端
- Amazon IAM Identity Center 不适用于您的账户，而且您没有其他身份提供商

创建用于紧急访问的 IAM 用户

IAM 用户是 Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。

让 IAM 用户进行紧急访问是创建 IAM 用户的推荐理由之一，这样当您的身份提供商无法访问时，您可以访问您的 Amazon Web Services 账户。

> **Note**
>
> 作为安全最佳实践，我们建议您通过身份联合验证而非创建 IAM 用户来提供对资源的访问权限。要了解需要使用 IAM 用户的特定情况，请参阅 何时创建 IAM 用户（而非角色）。

创建用于紧急访问的 IAM 用户

> **最小权限**
>
> 要执行下列步骤，您必须至少具有以下 IAM 权限：
>
> - `access-analyzer:ValidatePolicy`
> - `iam:AddUserToGroup`
> - `iam:AttachGroupPolicy`
> - `iam:CreateGroup`
> - `iam:CreateLoginProfile`
> - `iam:CreateUser`
> - `iam:GetAccountPasswordPolicy`
> - `iam:GetLoginProfile`
> - `iam:GetUser`
> - `iam:ListAttachedGroupPolicies`
> - `iam:ListAttachedUserPolicies`
> - `iam:ListGroupPolicies`
> - `iam:ListGroups`
> - `iam:ListGroupsForUser`
> - `iam:ListPolicies`
> - `iam:ListUserPolicies`
> - `iam:ListUsers`

Console

1. 按照Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。

2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择用户，然后选择创建用户。

> **Note**
>
> 如果您启用了 IAM Identity Center，则 Amazon Web Services 管理控制台 会显示一条提醒，提示您最好在 IAM Identity Center 中管理用户的访问权限。在此过程中，您创建的 IAM 用户专门在您的身份提供商不可用时使用。

4. 在指定用户详细信息页面中的用户详细信息下的用户名中，输入新用户的名称。这是 Amazon 的登录名。在此示例中，请输入 **`EmergencyAccess`**。

> **Note**
>
> 用户名可以是一个最多由 64 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您不能创建名为 TESTUSER 和 testuser 的两个用户。在策略中使用用户名或将其作为 ARN 的一部分时，用户名区分大小写。在控制台中向客户显示用户名时（例如在登录过程中），用户名不区分大小写。

5. 选择向 Amazon Web Services 管理控制台 提供用户访问权限 – 可选旁边的复选框，然后选择我想创建一个 IAM 用户。
6. 在控制台密码下，选择自动生成的密码。
7. 清除用户必须在下次登录时创建新密码（推荐）旁边的复选框。由于该 IAM 用户用于紧急访问，因此受信任的管理员会保留密码，仅在需要时提供密码。
8. 在设置权限页面上的权限选项下，选择将用户添加到组。然后，在用户组下，选择创建组。
9. 在创建用户组页面上的用户组名称中，输入 **`EmergencyAccessGroup`**。然后，在权限策略下，选择 AdministratorAccess。
10. 选择创建用户组，返回设置权限页面。
11. 在用户组下，选择您之前创建的 **`EmergencyAccessGroup`** 名称。
12. 选择下一步，进入查看并创建页面。
13. 在查看和创建页面上，查看要添加到新用户的用户组成员资格列表。如果您已准备好继续，请选择创建用户。
14. 在找回密码页面上，选择下载 .csv 文件以保存包含用户凭证信息（连接 URL、用户名和密码）的 .csv 文件。

15. 保存此文件，以便在您需要登录 IAM 且无权访问身份提供商时使用。

新的 IAM 用户显示在用户列表中。选择用户名称链接以查看用户详细信息。

Amazon CLI

1. 创建一个名为 **EmergencyAccess** 的用户。

   - aws iam create-user

```
aws iam create-user \
   --user-name EmergencyAccess
```

2. （可选）向用户提供对 Amazon Web Services 管理控制台的访问权限。这需要密码。要为 IAM 用户创建密码，您可以使用 `--cli-input-json` 参数传递包含密码的 JSON 文件。您还必须向用户提供账户登录页面的 URL。

   - aws iam create-login-profile

```
aws iam create-login-profile \
   --generate-cli-skeleton > create-login-profile.json
```

   - 在文本编辑器中打开 `create-login-profile.json` 文件并输入符合密码策略的密码，然后保存该文件。例如：

```
{
 "UserName": "EmergencyAccess",
 "Password": "Ex@3dRA0djs",
 "PasswordResetRequired": false
}
```

   - 再次使用 `aws iam create-login-profile` 命令，传递 `--cli-input-json` 参数以指定您的 JSON 文件。

```
aws iam create-login-profile \
```

```
   --cli-input-json file://create-login-profile.json
```

> **Note**
>
> 如果您在 JSON 文件中提供的密码违反了账户的密码策略，则将收到 `PasswordPolicyViolation` 错误。如果发生这种情况，则请查看您账户的密码策略，并更新 JSON 文件中的密码以符合要求。

3. 创建 **`EmergencyAccessGroup`**，将 Amazon 托管策略 `AdministratorAccess` 附加到组，然后将 **`EmergencyAccess`** 用户添加到该组。

> **Note**
>
> Amazon 托管策略 是由 Amazon 创建和管理的独立策略。每个策略都有自己的 Amazon 资源名称（ARN），其中包含策略名称。例如，`arn:aws:iam::aws:policy/IAMReadOnlyAccess` 是一个 Amazon 托管策略。有关 ARN 的更多信息，请参阅 IAM ARN。有关适用于 Amazon Web Services 服务 的 Amazon 托管式策略的列表，请参阅 Amazon 托管式策略。

- aws iam create-group

```
aws iam create-group \
   --group-name EmergencyAccessGroup
```

- aws iam attach-group-policy

```
aws iam attach-group-policy \
   --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
   --group-name >EmergencyAccessGroup
```

- aws iam add-user-to-group

```
aws iam add-user-to-group \
   --user-name EmergencyAccess \
   --group-name EmergencyAccessGroup
```

- 运行 aws iam get-group 命令列出 **EmergencyAccessGroup** 及其成员。

```
aws iam get-group \
   --group-name EmergencyAccessGroup
```

为无法使用 IAM 角色的工作负载创建 IAM 用户

Important

作为最佳实践，建议您要求您的人类用户在访问 Amazon 时使用临时凭证。
或者，您可以使用 Amazon IAM Identity Center 管理自己的用户身份，包括您的管理用户。建议使用 IAM Identity Center 来管理对您账户的访问以及这些账户中的权限。如果您使用的是外部身份提供商，则还可以在 IAM Identity Center 中配置用户身份的访问权限。

如果您的使用案例需要具有编程访问权限和长期凭证的 IAM 用户，我们建议您确立程序以在需要时更新访问密钥。有关更多信息，请参阅 更新访问密钥。

要执行一些账户和服务管理任务，您必须使用根用户凭证登录。要查看需要您以根用户身份登录的任务，请参阅需要根用户凭证的任务。

为无法使用 IAM 角色的工作负载创建 IAM 用户

最小权限

要执行下列步骤，您必须至少具有以下 IAM 权限：

- iam:AddUserToGroup
- iam:AttachGroupPolicy
- iam:CreateAccessKey
- iam:CreateGroup
- iam:CreateServiceSpecificCredential
- iam:CreateUser
- iam:GetAccessKeyLastUsed

- iam:GetAccountPasswordPolicy
- iam:GetAccountSummary
- iam:GetGroup
- iam:GetLoginProfile
- iam:GetPolicy
- iam:GetRole
- iam:GetUser
- iam:ListAccessKeys
- iam:ListAttachedGroupPolicies
- iam:ListAttachedUserPolicies
- iam:ListGroupPolicies
- iam:ListGroups
- iam:ListGroupsForUser
- iam:ListInstanceProfilesForRole
- iam:ListMFADevices
- iam:ListPolicies
- iam:ListRoles
- iam:ListRoleTags
- iam:ListSSHPublicKeys
- iam:ListServiceSpecificCredentials
- iam:ListSigningCertificates
- iam:ListUserPolicies
- iam:ListUserTags
- iam:ListUsers
- iam:UploadSSHPublicKey
- iam:UploadSigningCertificate

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。

2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择用户，然后选择创建用户。
4. 在指定用户详细信息页面中，执行以下操作：
   a. 对于 User name，键入 ***WorkloadName***。将 ***WorkloadName*** 替换为要使用该账户的工作负载的名称。
   b. 选择下一步。
5. （可选）在设置权限页面上，执行以下操作：
   a. 选择 Add user to group。
   b. 选择创建群组。
   c. 在创建用户组对话框中，对于用户组名称，键入代表组中工作负载使用情况的名称。在本示例中，使用名称 **Automation**。
   d. 在权限策略下，选中 PowerUserAccess 托管策略对应的复选框。

      > **Tip**
      >
      > 在权限策略搜索框中输入 Power 以快速找到此托管策略。

   e. 选择创建用户组。
   f. 返回到带有 IAM 组列表的页面，选中新用户组所对应的复选框。如果列表中未显示新组，请选择 Refresh（刷新）。
   g. 选择下一步。
6. （可选）在标签部分中，通过以键值对的形式附加标签来向用户添加元数据。有关更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
7. 验证新用户的用户组成员资格。如果您已准备好继续，请选择 Create user（创建用户）。
8. 此时将显示一条状态通知，告知您已成功创建用户。选择查看用户，进入用户详细信息页面
9. 选择安全凭证选项卡。然后，创建工作负载所需的凭证。
   - 访问密钥 - 选择创建访问密钥，为用户生成和下载访问密钥。

> **Important**
>
> 这是您查看或下载秘密访问密钥的唯一机会，您必须向用户提供此信息，他们才能使用 Amazon API。将用户的新访问密钥 ID 和秘密访问密钥保存在安全的地方。完成此步骤后，您再也无法访问这些秘密访问密钥。

- 用于 Amazon CodeCommit 的 SSH 公有密钥 – 选择上传 SSH 公有密钥上传 SSH 公有密钥，这样用户就可以通过 SSH 与 CodeCommit 存储库进行通信。
- 用于 Amazon CodeCommit 的 HTTPS Git 凭证 – 选择生成凭证以生成一组与 Git 存储库配合使用的唯一用户凭证。选择下载凭证，将用户名和密码保存到 .csv 文件中。这是该信息可用的唯一时间。如果您忘记或丢失密码，则需要重置密码。
- 用于 Amazon Keyspaces（Apache Cassandra 兼容）的凭证 - 选择生成凭证以生成用于 Amazon Keyspaces 的特定服务用户凭证。选择下载凭证，将用户名和密码保存到 .csv 文件中。这是该信息可用的唯一时间。如果您忘记或丢失密码，则需要重置密码。

> **Important**
>
> 服务特定凭证是与特定 IAM 用户相关联的长期凭证，只能用于所针对的服务。要向 IAM 角色或联合身份授予使用临时凭证访问所有 Amazon 资源的权限，应将 Amazon 身份验证与适用于 Amazon Keyspaces 的 SigV4 身份验证插件搭配使用。有关更多信息，请参阅 Amazon Keyspaces（Apache Cassandra 兼容）开发人员指南中的使用临时凭证连接到使用 IAM 角色和 SigV4 插件的 Amazon Keyspaces（Apache Cassandra 兼容）。

- X.509 签名证书 - 如果您需要发出安全的 SOAP 协议请求且位于 Amazon Certificate Manager 不支持的区域，请选择创建 X.509 证书。ACM 是预置、管理和部署您的服务器证书的首选工具。有关使用 ACM 的更多信息，请参阅 Amazon Certificate Manager 用户指南。

您已经创建具有编程访问权限的用户，并且使用 PowerUserAccess 任务函数对其进行配置。该用户的权限策略授予对除 IAM 和 Amazon Organizations 之外的所有服务的完全访问权限。

如果工作负载无法担任 IAM 角色，则可以使用此相同的流程向其他工作负载授予对您 Amazon Web Services 账户 资源的编程访问权限。此过程使用 PowerUserAccess 托管策略来分配权限。要遵循最低权限的最佳实践，可以考虑使用限制性更强的策略或创建自定义策略，将访问权限限制为仅访问该计划所需的资源。要了解有关使用限制用户对特定 Amazon 资源的权限的策略的信息，请

参阅适用于 Amazon 资源的 Access Management和IAM 基于身份的策略示例。要在创建用户组之后向其中添加其他用户，请参阅 编辑 IAM 组中的用户。

Amazon CLI

1. 创建一个名为 **Automation** 的用户。

   - aws iam create-user

   ```
   aws iam create-user \
       --user-name Automation
   ```

2. 创建一个名为 **AutomationGroup** 的 IAM 用户组，将 Amazon 托管策略 PowerUserAccess 附加到该组，然后将 **Automation** 用户添加到该组。

   > **Note**
   >
   > Amazon 托管策略 是由 Amazon 创建和管理的独立策略。每个策略都有自己的 Amazon 资源名称（ARN），其中包含策略名称。例如，arn:aws:iam::aws:policy/IAMReadOnlyAccess 是一个 Amazon 托管策略。有关 ARN 的更多信息，请参阅 IAM ARN。有关适用于 Amazon Web Services 服务 的 Amazon 托管式策略的列表，请参阅 Amazon 托管式策略。

   - aws iam create-group

   ```
   aws iam create-group \
       --group-name AutomationGroup
   ```

   - aws iam attach-group-policy

   ```
   aws iam attach-group-policy \
       --policy-arn arn:aws:iam::aws:policy/PowerUserAccess \
       --group-name AutomationGroup
   ```

   - aws iam add-user-to-group

```
aws iam add-user-to-group \
    --user-name Automation \
    --group-name AutomationGroup
```

- 运行 aws iam get-group 命令列出 **AutomationGroup** 及其成员。

```
aws iam get-group \
    --group-name AutomationGroup
```

3. 创建工作负载所需的安全凭证。

- 创建用于测试的访问密钥：aws iam create-access-key

```
aws iam create-access-key \
    --user-name Automation
```

此命令的输出显示秘密访问密钥和访问密钥 ID。将此信息记录并存储在安全的位置。如果这些凭证丢失，将无法恢复，则必须创建一个新的访问密钥。

> **Important**
>
> 这些 IAM 用户访问密钥是长期凭证，会给您的账户带来安全风险。完成测试后，建议您删除这些访问密钥。如果您有考虑访问密钥的场景，则请调查是否可以为工作负载 IAM 用户启用 MFA，并使用 aws sts get-session-token 获取会话的临时凭证，而不是使用 IAM 访问密钥。

- 上传 Amazon CodeCommit 的 SSH 公有密钥：aws iam upload-ssh-public-key

以下示例假设您的 SSH 公有密钥存储在文件 `sshkey.pub` 中。

```
aws upload-ssh-public-key \
    --user-name Automation \
    --ssh-public-key-body file://sshkey.pub
```

- 上传 X.509 签名证书：aws iam upload-signing-certificate

  如果您需要发出安全 SOAP 协议请求且位于 Amazon Certificate Manager 不支持的区域，则请上传 X.509 证书。ACM 是预置、管理和部署您的服务器证书的首选工具。有关使用 ACM 的更多信息，请参阅 Amazon Certificate Manager 用户指南。

  以下示例假设您的 X.509 签名证书存储在文件 `certificate.pem` 中。

  ```
  aws iam upload-signing-certificate \
  --user-name Automation \
  --certificate-body file://certificate.pem
  ```

如果工作负载无法担任 IAM 角色，则可以使用此相同的流程向其他工作负载授予对您 Amazon Web Services 账户 资源的编程访问权限。此过程使用 PowerUserAccess 托管策略来分配权限。要遵循最低权限的最佳实践，可以考虑使用限制性更强的策略或创建自定义策略，将访问权限限制为仅访问该计划所需的资源。要了解有关使用限制用户对特定 Amazon 资源的权限的策略的信息，请参阅适用于 Amazon 资源的 Access Management和IAM 基于身份的策略示例。要在创建用户组之后向其中添加其他用户，请参阅 编辑 IAM 组中的用户。

将多重身份验证用于身份验证

将多重身份验证（MFA）用于身份验证是另一项 IAM 最佳实践建议。MFA 是一个额外的安全层，它要求用户在提供用户名和密码以验证其身份后提供额外的身份验证因素。该功能显著增强了安全性，使攻击者更难获得未经授权的访问，即使用户的密码遭到泄露也是如此。MFA 被广泛用作保护在线账户、云服务和其他敏感资源访问的最佳实践。Amazon 支持根用户、IAM 用户、IAM Identity Center 的用户、Builder ID 和联合用户的 MFA。为提高安全性，您可以创建策略，要求在允许用户访问资源或采取特定操作之前配置 MFA，并将这些策略附加到您的 IAM 角色。IAM Identity Center 已预先配置为默认开启 MFA，因此 IAM Identity Center 中的所有用户除了使用用户名和密码外，还必须使用 MFA 登录。

> **Note**
>
> 所有 Amazon Web Services 账户 账户类型（独立账户、管理账户和成员账户）都需要为其根用户配置 MFA。如果尚未启用 MFA，则用户必须在首次登录尝试访问 Amazon Web Services 管理控制台后 35 天内注册 MFA。

有关更多信息，请参阅 Configure MFA in IAM Identity Center 和 IAM 中的 Amazon 多重身份验证。

Amazon Identity and Access Management 中的安全最佳实践和使用案例

Amazon Identity and Access Management（IAM）提供了您在开发和实施自己的安全策略时需要考虑的多项安全功能。以下最佳实操是一般准则，并不代表完整的安全解决方案。

这些最佳实践可能不适合或不足以满足您的特定环境。因此，请将它们视为有用的注意事项而不是要求。

要从 IAM 获取最大收益，需要花一些时间了解建议的最佳实践。完成这项工作的有效方法之一是了解如何在实际方案中结合使用 IAM 和其他 Amazon Web Services 服务。

主题

- IAM 的安全防御最佳实操
- 您的 Amazon Web Services 账户 的根用户最佳实践
- IAM 的商用案例

IAM 的安全防御最佳实操

Follow us on Twitter

为了帮助您确保 Amazon 资源的安全，请遵循 Amazon Identity and Access Management（IAM）的这些最佳实践。

主题

- 要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon
- 要求工作负载使用具有 IAM 角色的临时凭证访问 Amazon
- 需要多重身份验证（MFA）
- 对于需要长期凭证的用例，应在需要时更新访问密钥
- 遵循最佳实践以保护根用户凭证
- 应用最低权限权限
- Amazon 托管策略及转向最低权限权限入门
- 使用 IAM Access Analyzer 根据访问活动生成最低权限策略
- 定期查看并移除未使用的用户、角色、权限、策略和凭证
- 使用 IAM policy 中的条件进一步限制访问权限

- 使用 IAM Access Analyzer 验证对资源的公共和跨账户访问
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性
- 跨多个账户建立权限护栏机制
- 使用权限边界在账户内委派权限管理

要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon

人类用户，也称为人类身份，是应用程序的人员、管理员、开发人员、操作员和使用者。他们必须有身份才能访问您的 Amazon 环境和应用程序。作为组织成员的人类用户也称为员工身份。人类用户也可以是您与之协作以及与您的 Amazon 资源交互的外部用户。他们可以通过 Web 浏览器、客户端应用程序、移动应用程序或交互式命令行工具执行此操作。

请要求您的人类用户在访问 Amazon 时使用临时凭证。您可以使用身份提供程序来以代入角色的方式，为人类用户提供对 Amazon Web Services 账户 的联合访问权限，这样将提供临时凭证。对于集中式访问权限管理，我们建议使用 Amazon IAM Identity Center（IAM Identity Center）来管理对您账户的访问权限以及这些账户中的其他权限。您可以使用 IAM Identity Center 管理您的用户身份，或者从外部身份提供者管理 IAM Identity Center 中用户身份的访问权限。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 Amazon IAM Identity Center。

有关角色的更多信息，请参阅 角色术语和概念。

要求工作负载使用具有 IAM 角色的临时凭证访问 Amazon

工作负载是一系列资源和代码，它们可提供商业价值，如应用程序或后端过程。您的工作负载可能包含应用程序、操作工具和组件，它们需要凭证才能向 Amazon Web Services 服务 发送请求，例如请求从 Amazon S3 读取数据。

当您在 Amazon 计算服务（例如 Amazon EC2 或 Lambda）上进行构建时，Amazon 会向该计算资源提供 IAM 角色的临时凭证。使用 Amazon SDK 编写的应用程序将发现并使用这些临时凭证来访问 Amazon 资源，并且无需将 IAM 用户的长期凭证分配给在 Amazon 上运行的工作负载。

在 Amazon 外部运行的工作负载，例如您的本地服务器、来自其他云提供商的服务器或托管式持续集成和持续交付（CI/CD）平台，仍可以使用临时凭证。但是，您需要将这些临时凭证提供给工作负载。以下是向工作负载提供临时凭证的方法：

- 您可以使用 IAM Roles Anywhere 通过公有密钥基础设施（PKI）的 X.509 凭证为工作负载申请临时 Amazon 凭证。

- 您可以使用在 Amazon Web Services 账户 内部配置的外部身份提供者（IdP）的 SAML 断言来调用 Amazon STS `AssumeRoleWithSAML` API，为工作负载请求临时 Amazon 凭证。
- 您可以调用 Amazon STS `AssumeRoleWithWebIdentity` API，使用在 Amazon Web Services 账户 中配置的 IdP 的 JSON 网络令牌（JWT）为您的工作负载请求临时 Amazon 凭证。
- 您可以使用 Amazon IoT Core 通过双向传输层安全 (MTLS) 身份验证向物联网设备请求临时 Amazon 凭证。

有些 Amazon Web Services 服务 还支持集成，以便为在 Amazon 之外运行的工作负载提供临时凭证：

- 借助 Amazon Elastic Container Service（Amazon ECS）Anywhere，您可在自己的计算资源上运行 Amazon ECS 任务，并为在这些计算资源上运行的 Amazon ECS 任务提供临时 Amazon 凭证。
- 借助 Amazon Elastic Kubernetes Service Hybrid Nodes，您能够将在 Amazon 之外作为节点运行的计算资源加入 Amazon EKS 集群。Amazon EKS 可以向在您的计算资源上运行的 Amazon EKS 容器组（pod）提供临时凭证。
- 借助 Amazon Systems ManagerHybrid Activations，您可以通过 SSM 管理在 Amazon 之外运行的计算资源，并为在您的计算资源上运行的 SSM 代理提供临时 Amazon 凭证。

需要多重身份验证（MFA）

我们建议对访问您 Amazon 资源的人类用户和工作负载使用 IAM 角色，以便他们使用临时凭证。但是，如果您的账户中需要 IAM 用户或根用户，则需要使用 MFA 来提高安全性。有了 MFA，用户就有了一个可以生成身份验证质询响应的设备。各个用户的凭证和设备生成的响应是完成登录过程所必需的。有关更多信息，请参阅 IAM 中的 Amazon 多重身份验证。我们建议您尽可能使用防网络钓鱼的 MFA，例如通行密钥和安全密钥。有关更多信息，请参阅 在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥。

如果您使用 IAM Identity Center 来对人类用户进行集中访问管理，则当您的身份源配置了 IAM Identity Center 身份存储、Amazon Managed Microsoft AD 或 AD Connector 时，您可以使用 IAM Identity Center MFA 功能。有关 IAM Identity Center 中的 MFA 的更多信息，请参阅Amazon IAM Identity Center《用户指南》中的多重身份验证。

对于需要长期凭证的用例，应在需要时更新访问密钥

在可能的情况下，我们建议使用临时凭证，而不是创建访问密钥等长期凭证。但对于需要具有编程访问权限和长期凭证的 IAM 用户的场景，我们建议在需要时更新访问密钥，例如在员工从公司离职时。我

们建议利用上次使用 IAM 访问信息来安全地更新和移除访问密钥。有关更多信息，请参阅 更新访问密钥。

在 Amazon 中，有一些特定的使用场景需要带有 IAM 用户的长期凭证。部分使用场景包括：

- 无法使用 IAM 角色的工作负载 – 您可以从需要访问 Amazon 的位置运行工作负载。在某些应用场景中，您无法使用 IAM 角色提供临时凭证，例如 WordPress 插件。在这些情况下，请使用该工作负载的 IAM 用户长期访问密钥对 Amazon 进行身份验证。
- 第三方 Amazon 客户端 – 如果您正在使用的工具不支持访问 IAM Identity Center，如不在 Amazon 上托管的第三方 Amazon 客户端或供应商，请使用 IAM 用户长期访问密钥。
- Amazon CodeCommit 访问 – 如果您正在使用 CodeCommit 来存储您的代码，则可以使用具有 SSH 密钥或服务特定凭证的 IAM 用户对 CodeCommit 进行存储库身份验证。我们建议您除了使用 IAM Identity Center 中的用户进行普通身份验证之外，再执行此操作。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon Web Services 账户 或云应用程序的人员。要在不配置 IAM 用户的情况下授予用户访问您 CodeCommit 存储库的权限，您可以配置 git-remote-codecommit 实用工具。有关 IAM 和 CodeCommit 的更多信息，请参阅 CodeCommit 的 IAM 凭证：Git 凭证、SSH 密钥和 Amazon 访问密钥。有关配置 git-remote-codecommit 实用工具的更多信息，请参阅《Amazon CodeCommit 用户指南》中的连接到具有轮换凭证的 Amazon CodeCommit 存储库。
- Amazon Keyspaces (for Apache Cassandra) access（Amazon Keyspaces（Apache Cassandra 兼容）访问）– 在您无法使用 IAM Identity Center 中的用户的情况下，如出于测试 Cassandra 兼容性的目的，您可以使用具有服务特定凭证的 IAM 用户向 Amazon Keyspaces 进行身份验证。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon Web Services 账户 或云应用程序的人员。您还可以使用临时凭证连接到 Amazon Keyspaces。有关更多信息，请参阅《Amazon Keyspaces（Apache Cassandra 兼容）开发人员指南》中的使用临时凭证连接到使用 IAM 角色和 SigV4 插件的 Amazon Keyspaces。

遵循最佳实践以保护根用户凭证

在创建 Amazon Web Services 账户 时，您将创建根用户凭证，以登录 Amazon Web Services 管理控制台。请像保护其他敏感个人信息一样保护您的根用户凭证。要更好地了解如何保护和扩展根用户进程，请参阅 您的 Amazon Web Services 账户 的根用户最佳实践。

应用最低权限权限

在使用 IAM policy 设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限权限。在探索工作负载或使用场景所需的权限时，您可以从较

广泛的权限开始。随着使用场景逐渐成熟，您可以努力减少授予权限，直至达到最低权限的标准。有关使用 IAM 应用权限的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限。

Amazon 托管策略及转向最低权限权限入门

要开始向用户和工作负载授予权限，请使用 Amazon 托管策略来为许多常见使用场景授予权限。您可以在 Amazon Web Services 账户 中找到这些策略。请记住，Amazon 托管策略可能不会为您的特定使用场景授予最低权限权限，因为它们可供所有 Amazon 客户使用。因此，我们建议通过定义特定于您的使用场景的客户管理型策略来减少权限。有关更多信息，请参阅 Amazon托管策略。有关专为特定任务函数制定的 Amazon 托管策略的更多信息，请参阅 工作职能的 Amazon 托管策略。

使用 IAM Access Analyzer 根据访问活动生成最低权限策略

如果仅授予执行任务所需的权限，您可以根据记录在 Amazon CloudTrail 中的访问活动生成策略。IAM Access Analyzer 会分析您的 IAM 角色使用的服务和操作，然后生成您可以使用的精细策略。测试每个生成的策略后，可以将该策略部署到生产环境中。这可确保您仅向工作负载授予所需的权限。有关策略生成的更多信息，请参阅 IAM Access Analyzer 策略生成。

定期查看并移除未使用的用户、角色、权限、策略和凭证

您的 Amazon Web Services 账户 中可能存在不再需要的 IAM 用户、角色、权限、策略或凭证。IAM 提供了上次访问的信息来帮助您识别不再需要的用户、角色、权限、策略和凭证，以便您将其删除。这有助于减少必须监控的用户、角色、权限、策略和凭证的数量。您可以使用此信息优化 IAM policy 以更好地遵循最低权限权限。有关更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

使用 IAM policy 中的条件进一步限制访问权限

您可以指定策略语句在何种条件下生效。这样，您就可以授予对操作和资源的访问权限，但前提是访问请求满足特定条件。例如，您可以编写策略条件来指定必须使用 TLS 发送所有请求。您也可以使用条件来授予对服务操作的访问权限，但前提是它们是通过特定的 Amazon Web Services 服务 使用的，例如 Amazon CloudFormation。有关更多信息，请参阅 IAM JSON 策略元素：Condition。

使用 IAM Access Analyzer 验证对资源的公共和跨账户访问

在 Amazon 中授予公共或跨账户访问权限之前，我们建议您验证是否需要此类访问权限。您可以使用 IAM Access Analyzer 来帮助您预览和分析对受支持资源类型的公共和跨账户访问。您可以通过查看 IAM Access Analyzer 生成的结果来完成此操作。这些结果可帮助您验证资源访问控制是否授予了期望的访问权限。此外，在更新公共和跨账户权限时，您可以在对资源部署新的访问控制之前验证更改的效

果。IAM Access Analyzer 还会持续监控受支持的资源类型，并为允许公共或跨账户访问的资源生成结果。有关更多信息，请参阅使用 IAM Access Analyzer API 预览访问权限。

使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性

验证您创建的策略以确保它们符合 IAM policy 语言（JSON）和 IAM 最佳实践。您可以使用 IAM Access Analyzer 策略验证来验证您的策略。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。在控制台中创作新策略或编辑现有策略时，IAM Access Analyzer 会提供一些建议，帮助您在保存策略之前对其进行优化和验证。此外，我们还建议您查看和验证所有现有策略。有关更多信息，请参阅 IAM Acess Analyzer 策略验证。有关 IAM Access Analyzer 提供的策略检查的更多信息，请参阅 IAM Access Analyzer 策略检查参考。

跨多个账户建立权限护栏机制

在扩展工作负载时，请使用由 Amazon Organizations 托管的多个账户将工作负载分离。建议您使用 Amazon Organizations 服务控制策略（SCP）建立权限护栏，以控制账户中所有主体（IAM 角色和用户）的访问权限。建议您使用 Amazon Organizations 资源控制策略（RCP）建立权限护栏，以控制组织中 Amazon 资源的访问权限。SCP 和 RCP 是组织策略类型，可用于在 Amazon 组织、组织单元（OU）或账户级别管理组织中的权限。

然而，单凭 SCP 和 RCP 不足以授予您组织中的主体和资源权限。SCP 和 RCP 不授予任何权限。要授予权限，您必须将基于身份或基于资源的策略附加到 IAM 用户、IAM 角色或者您账户中的资源。有关更多信息，请参阅 SRA 构建块 – Amazon Organizations、账户和护栏。

使用权限边界在账户内委派权限管理

在某些场景中，您可能需要将账户中的权限管理委派给其他用户。例如，您可以允许开发人员为其工作负载创建和管理角色。将权限委派给其他人时，请使用权限边界以设置您委派的最大权限数量。权限边界是一个高级功能，用于使用托管策略设置基于身份的策略可授予 IAM 角色的最大权限。权限边界自己不授予访问权限。有关更多信息，请参阅 IAM 实体的权限边界。

您的 Amazon Web Services 账户 的根用户最佳实践

首次创建 Amazon Web Services 账户 时，从一组可以完全访问您账户中的所有 Amazon 资源的默认凭证开始。此身份称作 Amazon Web Services 账户 根用户。强烈建议您不要访问 Amazon Web Services 账户 根用户，除非您有一项需要根用户凭证的任务。您需要保护您的根用户凭证和账户恢复机制的安全，以帮助确保您的高权限凭证不会泄露给未经授权的使用。

对于通过 Amazon Organizations 管理的多个 Amazon Web Services 账户，建议从成员账户中移除根用户凭证，以帮助防止未经授权的使用。您可以移除根用户密码、访问密钥、签名证书，并停用和删除多重身份验证（MFA）。成员账户不能登录到他们的根用户或为其根用户执行密码恢复。有关更多信息，请参阅 集中管理成员账户的根访问权限。

与其访问根用户，不如创建管理用户来处理日常任务。

- 有关单个独立的 Amazon Web Services 账户，请参阅 保护 IAM 用户。
- 有关通过 Amazon Organizations 托管的多个 Amazon Web Services 账户，请参阅为 IAM Identity Center 管理用户设置 Amazon Web Services 账户 访问权限。

然后，您可以使用您的管理用户为需要访问您 Amazon Web Services 账户 中资源的用户创建其他身份。强烈建议您要求用户在访问 Amazon 时使用临时凭证进行身份验证。

- 对于单个独立的 Amazon Web Services 账户，请使用 IAM 角色 在您的账户中创建具有特定权限的身份。角色旨在让任何需要它的用户代入。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。与 IAM 角色不同，IAM 用户 具有密码和访问密钥等长期凭证。如有可能，最佳实践建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。
- 对于通过 Amazon Organizations 托管的多个 Amazon Web Services 账户，请使用 IAM Identity Center 员工用户。借助 IAM Identity Center，您可以集中管理您 Amazon Web Services 账户 中的用户以及这些账户内的权限。使用 IAM Identity Center 或通过外部身份提供商来管理您的用户身份。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 Amazon IAM Identity Center。

主题

- 保护您的根用户凭证，以防止未经授权的使用
- 使用强根用户密码来加强访问保护
- 使用多重身份验证（MFA）保护您的根用户登录安全
- 请勿为根用户创建访问密钥
- 尽可能使用多人审批进行根用户登录
- 将组电子邮件地址用于根用户凭证
- 限制对账户恢复机制的访问
- 保护您的 Amazon Organizations 账户的根用户凭证
- 监控访问权限和使用情况

保护您的根用户凭证，以防止未经授权的使用

保护您的根用户凭证安全，并将其仅用于需要它们的任务。为了帮助防止未经授权的使用，请勿与任何人共享您的根用户密码、MFA、访问密钥、CloudFront 密钥对或签名证书，因有严格业务需求而访问根用户的人员除外。

请勿使用依赖 Amazon Web Services 服务 的工具将根用户密码存储在使用相同密码访问的账户中。如果丢失或忘记了根用户密码，则将无法访问这些工具。建议您优先考虑弹性，并考虑要求两个或以上的人授权对存储位置的访问。对密码或其存储位置的访问都应有日志记录和监控。

使用强根用户密码来加强访问保护

我们建议您使用独有的强密码。诸如具有强大密码生成算法的密码管理器之类的工具可帮助您实现这些目标。Amazon 要求您的密码符合以下条件：

- 长度必须至少 8 个字符，最多 128 个字符。
- 必须至少包含以下字符类型中三种的组合：大写字母、小写字母、数字，以及 ! @ # $ % ^ & * () <> [] {} | _+-= 符号。
- 不得与您的 Amazon Web Services 账户名称或电子邮件地址相同。

有关更多信息，请参阅 更改Amazon Web Services 账户根用户密码。

使用多重身份验证（MFA）保护您的根用户登录安全

由于根用户可以执行特权操作，因此除了电子邮件地址和密码作为登录凭证之外，还必须为根用户添加 MFA 作为第二个身份验证因素。您最多可以向 Amazon Web Services 账户 根用户注册 8 台当前支持的 MFA 类型的任意组合的 MFA 设备。

强烈建议为您的根用户凭证启用多个 MFA，以便为您的安全策略提供更大的灵活性和弹性。所有 Amazon Web Services 账户 账户类型（独立账户、管理账户和成员账户）都需要为其根用户配置 MFA。如果尚未启用 MFA，则用户必须在首次登录尝试访问 Amazon Web Services 管理控制台后 35 天内注册 MFA。

- FIDO 认证的硬件安全密钥由第三方提供商提供。有关更多信息，请参阅为 Amazon Web Services 账户 根用户启用 FIDO 安全密钥。
- 一种以基于时间的一次性密码（TOTP）算法为基础生成六位数字代码的硬件设备。有关更多信息，请参阅为 Amazon Web Services 账户 根用户启用硬件 TOTP 令牌。

- 在电话或其他设备上运行并模拟物理设备的虚拟身份验证器应用程序。有关更多信息，请参阅为 Amazon Web Services 账户 根用户启用虚拟 MFA 设备。

请勿为根用户创建访问密钥

访问密钥允许您在 Amazon 命令行界面（Amazon CLI）中运行命令或使用其中一个 Amazon SDK 中的 API 操作。强烈建议您不要为根用户创建访问密钥，因为根用户对账户中的所有 Amazon Web Services 服务和资源拥有完全访问权限，包括账单信息。

由于只有少数任务需要根用户，而且您通常不经常执行这些任务，因此我们建议登录到 Amazon Web Services 管理控制台 来执行根用户任务。对于编程访问，无需为根用户创建访问密钥，而是使用带有根凭证的 `aws login` 命令对 Amazon CLI 和 SDK 进行身份验证。此方法提供临时、自动轮换的凭证，可增强您的安全状况，同时无需管理长期访问密钥。在创建访问密钥之前，请查看 长期访问密钥的替代方案。

尽可能使用多人审批进行根用户登录

请考虑使用多人审批，以确保没有人可以同时访问根用户的 MFA 和密码。一些公司通过设置一组具有密码访问权限的管理员和另一组具有 MFA 访问权限的管理员来增加额外的安全层。这两个组中必须各有一名成员以根用户身份登录。

将组电子邮件地址用于根用户凭证

使用您的业务管理并将收到的消息直接转发到一组用户的电子邮件地址。如果 Amazon 必须联系账户所有者，这种方法有助于降低响应延迟的风险，即使个人在度假、生病或离开公司时也是如此。根用户使用的电子邮件地址不得用于其他目的。

限制对账户恢复机制的访问

确保制定管理根用户凭证恢复机制的流程，以防在紧急情况（例如接管您的管理账户）下需要访问该机制。

- 确保您可以访问根用户电子邮件收件箱，以便可以重置丢失或忘记的根用户密码。
- 如果您 Amazon Web Services 账户 根用户的 MFA 丢失、损坏或无法正常工作，则可以使用已注册到同一根用户凭证的另一个 MFA 登录。如果失去了对所有 MFA 的访问权限，则需要将用于注册账户的电话号码和电子邮件保持最新和可访问状态，以便恢复 MFA。有关详细信息，请参阅恢复根用户 MFA 设备。

- 如果您选择不存储根用户密码和 MFA，则可以将在您的账户中注册的电话号码用作恢复根用户凭证的替代方式。确保您可以访问联系人电话号码，保持更新电话号码，并限制谁有权管理该电话号码。

任何人都不应同时访问电子邮件收件箱和电话号码，因为两者都是找回根用户密码的验证通道。让两组人管理这些通道很重要。一个组可以访问您的主电子邮件地址，另一个组可以访问主电话号码，从而以根用户身份恢复对您的账户的访问权限。

保护您的 Amazon Organizations 账户的根用户凭证

当您转向使用 Amazon Organizations 的多账户策略时，每个 Amazon Web Services 账户 都有自己的根用户凭证，您需要保护这些凭证。您用来创建组织的账户是管理账户，而组织中的其余账户是成员账户。

保护管理账户的安全根用户凭证

Amazon 要求您为组织的管理账户的根用户注册 MFA。MFA 注册必须在首次尝试登录期间或在 35 天宽限期内完成。如果在此时间内未启用 MFA，则需要先注册才能访问 Amazon Web Services 管理控制台。有关更多信息，请参阅 Amazon Web Services 账户根用户 的多重身份验证。

保护成员账户的安全根用户证书

如果您使用 Amazon Organizations 管理多个账户，则可以采用两种策略来保护 Amazon Organizations 中的根用户访问权限。

- 集中根访问并从成员账户中移除根用户凭证。移除根用户凭证、访问密钥、签名证书，并停用和删除多重身份验证（MFA）。使用此策略时，成员账户不能登录到他们的根用户或为其根用户执行密码恢复。有关更多信息，请参阅 集中管理成员账户的根访问权限。
- 使用 MFA 保护您的 Amazon Organizations 账户的根用户凭证，以增强账户安全性。有关更多信息，请参阅 Amazon Web Services 账户根用户 的多重身份验证。

有关详细信息，请参阅《Amazon Organizations 用户指南》中的访问组织中的成员账户。

使用服务控制策略（SCP）在 Amazon Organizations 中设置预防性安全控制

如果贵组织中的成员账户启用了根用户凭证，则您可以应用 SCP 来限制对成员账户根用户的访问。拒绝在您的成员账户中执行所有根用户操作（某些仅限根的操作除外）有助于防止未经授权的访问。有关详细信息，请参阅使用 SCP 限制成员账户中的根用户可以执行的操作。

监控访问权限和使用情况

建议您使用当前的跟踪机制来监控、提醒和报告根用户凭证的登录和使用情况，包括宣布根用户登录和使用情况的警报。以下服务可以帮助确保跟踪根用户凭证的使用情况，并进行安全检查，以帮助防止未经授权的使用。

> **Note**
>
> CloudTrail 日志记录了根用户和特权根用户会话的不同登录事件。这些特权会话允许在组织中的成员账户中执行需要根用户凭证的任务。您可以使用登录事件来标识管理账户或委派管理员使用 `sts:AssumeRoot` 所采取的操作。有关更多信息，请参阅 在 CloudTrail 中跟踪特权任务。

- 如果您想收到有关您账户中根用户登录活动的通知，则可以利用 Amazon CloudWatch 创建事件规则，该规则用于检测根用户凭证何时被使用及触发向您的安全管理员的通知。有关详细信息，请参阅 Amazon Web Services 账户 根用户活动的监控和通知。
- 如果您想设置通知以提醒您已批准的根用户操作，则可以利用 Amazon EventBridge 和 Amazon SNS 来编写 EventBridge 规则，以跟踪根用户对特定操作的使用情况，并使用 Amazon SNS 主题向您发出通知。有关示例，请参阅在创建 Amazon S3 对象时发送通知。
- 如果您已经使用 GuardDuty 作为威胁检测服务，则可以扩展其功能，使其能够在您的账户中使用根用户凭证时通知您。

提醒应包括但不限于用于根用户的电子邮件地址。实施响应提醒的程序，以便收到根用户访问提醒的人员了解如何验证是否需要根用户访问权限，以及在其认为发生安全事件时如何上报。有关如何配置提醒的示例，请参阅 Amazon Web Services 账户 根用户活动的监控和通知。

评估根用户是否符合 MFA

以下服务可帮助评估根用户凭证的 MFA 合规性。

> 如果您遵循移除根用户凭证的最佳实践，则与 MFA 相关的规则将返回不合规。
>
> 建议从贵组织的成员账户中移除根用户凭证，以帮助防止未经授权的使用。移除根用户凭证（包括 MFA）后，这些成员账户将被评估为不适用。

- Amazon Config 提供规则以监控对根用户最佳实践的遵守情况。您可以使用 Amazon Config 托管规则来帮助您对根用户凭证强制执行 MFA。Amazon Config 还可以识别根用户的访问密钥。
- Security Hub CSPM 提供您在 Amazon 中的安全状态的全面视图，可帮助您评测 Amazon 环境是否符合安全行业标准和最佳实践，例如对根用户进行 MFA 和不具有根用户访问密钥。有关可用规则的详细信息，请参阅《Security Hub CSPM User Guide》中的 Amazon Identity and Access Management controls。
- Trusted Advisor 提供安全检查，以便您知道根用户账户是否未启用 MFA。有关更多信息，请参阅《Amazon 支持用户指南》中的根账户上的 MFA。

如果您需要报告账户的安全问题，请参阅报告可疑电子邮件或漏洞报告。您还可以 联系 Amazon 以获取帮助和其他指导。

IAM 的商用案例

IAM 的简单商用案例可帮助您了解使用该服务来控制您的用户所拥有的 Amazon 访问权限的基本方法。此使用案例只是粗略介绍，并未涵盖您使用 IAM API 来实现所需结果的技术性细节。

此使用案例将探讨一家名为 Example Corp 的虚构公司可能使用 IAM 的两种典型方式。第一种场景考虑 Amazon Elastic Compute Cloud (Amazon EC2)。第二种场景考虑 Amazon Simple Storage Service (Amazon S3)。

有关通过 Amazon 的其他服务使用 IAM 的更多信息，请参阅 使用 IAM 的 Amazon 服务。

主题

- Example Corp 的初始设置
- 将 IAM 与 Amazon EC2 结合使用的使用案例
- 将 IAM 与 Amazon S3 结合使用的使用案例

Example Corp 的初始设置

Nikki Wolf 和 Mateo Jackson 是 Example Corp 的创始人。公司成立时，他们创建了一个 Amazon Web Services 账户 并建立了 Amazon IAM Identity Center（IAM Identity Center）来创建管理账户以使用他们的 Amazon 资源。当您为管理用户设置帐户访问权限时，IAM Identity Center 会创建相应的 IAM 角色。该角色由 IAM Identity Center 控制，在相关 Amazon Web Services 账户 中创建，并将在 AdministratorAccess 权限集中指定的策略附加到该角色。

由于他们现在拥有管理员账户，Nikki 和 Mateo 不再需要使用其根用户身份来访问他们的 Amazon Web Services 账户。他们计划仅将根用户用于完成仅限根用户开展的任务。在查看安全最佳实践之后，他们为根用户凭证配置多重身份验证（MFA），并决定如何保其护根用户凭证。

随着公司的发展，他们雇佣了员工，担任开发人员、管理员、测试人员、管理人员及系统管理员。Nikki 负责运营，而 Mateo 管理工程团队。他们建立了一个 Active Directory 域服务器来管理员工账户以及对公司内部资源的访问。

为了让员工访问 Amazon 资源，他们使用 IAM Identity Center 将公司的 Active Directory 连接到他们的 Amazon Web Services 账户。

由于他们将 Active Directory 连接到 IAM Identity Center，因此用户、组和组成员关系都是已同步和已定义的。他们必须为不同的组分配权限集和角色，以给予用户正确的 Amazon 资源访问级别。他们在 Amazon Web Services 管理控制台 中使用 工作职能的 Amazon 托管策略 来创建以下权限集：

- 管理员
- Billing
- 开发人员
- 网络管理员
- 数据库管理员
- 系统管理员
- 支持用户

然后，他们会将这些权限集分配给相应的角色，这些角色是分配至其 Active Directory 组的角色。

有关描述 IAM Identity Center 初始配置的分步指南，请参阅《Amazon IAM Identity Center 用户指南》中的入门。有关配置 IAM Identity Center 用户访问权限的更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的 Amazon 账户单点登录访问权限。

将 IAM 与 Amazon EC2 结合使用的使用案例

类似 Example Corp 的公司通常使用 IAM 与类似 Amazon EC2 的服务互动。如要理解使用案例的这一部分，您需要对 Amazon EC2 有基本的了解。有关 Amazon EC2 的更多信息，请转至《Amazon EC2 用户指南》。

用户组的 Amazon EC2 权限

为提供“周边”控制，Nikki 对 AllUsers 用户组附加了一个策略。如果来源 IP 地址位于 Example Corp 企业网络外部，则该策略拒绝用户的任何 Amazon 请求。

在 Example Corp，不同的 IAM 组需要不同的权限：

- System administrators - 需要创建和管理 AMI、实例、快照、卷、安全组等的权限。Nikki 向 SysAdmins 用户组附加了一个 `AmazonEC2FullAccess` Amazon 托管策略，以为该用户组成员授予使用所有 Amazon EC2 操作的权限。
- Developers（开发人员）- 只需能够使用实例即可。因此，Mateo 创建并向 Developers 用户组附加了策略，以允许开发人员调用 `DescribeInstances`、`RunInstances`、`StopInstances`、`StartInstances` 及 `TerminateInstances`。

> **Note**
>
> Amazon EC2 使用 SSH 密钥、Windows 密码及安全组来控制哪些人能够访问特定 Amazon EC2 实例的操作系统。在 IAM 系统中，无法允许或拒绝访问特定实例的操作系统。

- Support 用户 – 无法执行任何 Amazon EC2 操作，但可列出当前可用的 Amazon EC2 资源。因此，Nikki 创建并向 Support 用户组附加了一个策略，以便仅允许该组成员调用 Amazon EC2“Describe”API 操作。

如需有关上述策略具体形式的示例，请参阅《Amazon EC2 用户指南》中的 IAM 基于身份的策略示例和使用 Amazon Identity and Access Management。

用户的工作职能更改

此时，其中一位开发人员 Paulo Santos 的工作职能发生转变，成为一名管理人员。作为一名管理人员，Paulo 成为 Support 用户组的一员，这样他就可以为开发人员开立支持案例。Mateo 将 Paulo 从 Developers 用户组移至 Support 用户组。因此，他与 Amazon EC2 实例交互的能力受到了限制。他无法启动或启用实例。即使他是启动或启用实例的用户，也无法停止或终止现有实例。他只能列出 Example Corp 用户已启动的实例。

将 IAM 与 Amazon S3 结合使用的使用案例

类似 Example Corp 的公司通常还通过 Amazon S3 使用 IAM。John 已为公司创建一个名为 amzn-s3-demo-bucket 的 Amazon S3 存储桶。

创建其他用户和用户组

作为员工，Zhang Wei 和 Mary Major 都需要能够在公司的存储桶中创建自己的数据。他们还需要读取和写入所有开发人员都要处理的共享数据。为做到这一点，Mateo 采用 Amazon S3 键前缀方案，在 amzn-s3-demo-bucket 中按照逻辑方式排列数据，如下图所示。

```
/amzn-s3-demo-bucket
    /home
        /zhang
        /major
    /share
        /developers
        /managers
```

Mateo 针对每位员工将 /amzn-s3-demo-bucket 分隔成一系列主目录，并为开发人员和管理人员组留出一个共享区域。

现在，Mateo 创建一组策略，以便向用户和用户组分配权限：

- Zhang 的主目录访问 – Mateo 向 Wei 附加的策略允许后者读取、写入和列出任何带 Amazon S3 键前缀 /amzn-s3-demo-bucket/home/zhang/ 的对象
- Major 的主目录访问 – Mateo 向 Mary 附加的策略允许后者读取、写入和列出任何带 Amazon S3 键前缀 /amzn-s3-demo-bucket/home/major/ 的对象
- Developers 用户组的共享目录访问 – Mateo 向该用户组附加的策略允许开发人员读取、写入和列出 /amzn-s3-demo-bucket/share/developers/ 中的任何对象
- Managers 用户组的共享目录访问 – Mateo 向该用户组附加的策略允许管理人员读取、写入和列出 /amzn-s3-demo-bucket/share/managers/ 中的对象

> **Note**
>
> 对于创建存储桶或对象的用户，Amazon S3 不会自动授予其对存储桶或对象执行其他操作的权限。因此，在您的 IAM policy 中，您必须显式授予用户使用他们所创建的 Amazon S3 资源的权限。

有关这些策略具体形式的示例，请参阅 Amazon Simple Storage Service 用户指南中的访问控制。有关如何在运行时对策略进行评估的信息，请参阅策略评估逻辑。

用户的工作职能更改

此时，其中一位开发人员 Zhang Wei 的工作职能发生转变，成为一名管理人员。我们假设他不再需要访问 `share/developers` 目录中的文档。作为管理员，Mateo 将 Wei 从 `Developers` 用户组移至 `Managers` 用户组。通过简单的重新分配，Wei 将自动获得所有授予给 `Managers` 用户组的权限，但将无法再访问 `share/developers` 目录中的数据。

与第三方企业集成

组织经常与合作公司、顾问及承包商合作。Example Corp 是 Widget Company 的合作伙伴，而 Widget Company 的员工 Shirley Rodriguez 需要将数据放入存储桶中，以供 Example Corp 使用。Nikki 创建了一个名为 WidgetCo 的用户组和名为 `Shirley` 的用户，并将 Shirley 添加至 WidgetCo 用户组。Nikki 还创建了一个名为 amzn-s3-demo-bucket1 的专用存储桶以供 Shirley 使用。

Nikki 更新现有策略或添加新的策略来满足合作伙伴 Widget Company 的需求。例如，Nikki 可新建用于拒绝 WidgetCo 用户组成员使用任何操作（写入操作除外）的策略。除非有一个广泛的策略，授予所有用户访问大量 Amazon S3 操作的许可，否则此策略将非常必要。

IAM 教程

以下教程提供了 Amazon Identity and Access Management (IAM) 的常见任务的完整端到端过程。这些过程适用于实验室类型环境，使用了虚构的公司名称、用户名称等。其目的在于提供一般性指导。在未进行仔细审核并进行改进以满足组织环境的独特需求的情况下，这些过程不适合在您的生产环境中直接使用。

教程

- IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限
- IAM 教程：创建和附加您的第一个客户托管策略
- IAM 教程：根据标签定义访问 Amazon 资源的权限
- IAM 教程：允许用户管理其凭证和 MFA 设置
- IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者 (IdP)
- IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 联合身份验证 IAM 角色
- IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者（IdP）和 SAML 联合身份验证 IAM 角色

IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限

> **Important**
>
> IAM 最佳实践建议您您要求人类用户使用与身份提供商的联合身份验证才能使用临时凭证访问 Amazon，而不是使用具有长期凭证的 IAM 用户。建议您仅在联合用户不支持的特定用例中使用 IAM 用户。

本教程将指导您如何使用角色来委派对您拥有的不同 Amazon Web Services 账户（称为目标和原始）中资源的访问权限。您将与另一账户中的用户共享一个账户中的资源。通过以这种方式设置跨账户存取，您不需要在每个账户中创建单个 IAM 用户。此外，用户不必从一个账户注销并登录另一个账户，即可访问不同 Amazon Web Services 账户 中的资源。配置角色后，您将了解如何从 Amazon Web Services 管理控制台、Amazon CLI 和 API 中使用角色。

在本教程中，目标账户管理不同应用程序和团队访问的应用程序数据。在每个账户中，您将应用程序信息存储在 Amazon S3 存储桶中。您可以在原始账户中管理 IAM 用户，其中有两个 IAM 用户角色：开

发人员和分析师。开发人员和分析师使用原始账户生成由多个微服务共享的数据。两个角色都拥有在原始账户中工作的权限，可访问该账户中的资源。开发人员必须不时更新目标账户中的共享数据。开发人员将此数据存储在名为 amzn-s3-demo-bucket-shared-container 的 Amazon S3 存储桶中。

在本教程结束时，您将拥有以下内容：

- 原始账户（受信任账户）中能够担任目标账户中特定角色的用户。
- 目标账户（可信账户）中能够访问特定 Amazon S3 存储桶的角色。
- 目标账户中的 amzn-s3-demo-bucket-shared-container 存储桶。

开发人员可以在 Amazon Web Services 管理控制台 中使用该角色访问目标账户中的 amzn-s3-demo-bucket-shared-container 存储桶。他们还可以使用通过该角色提供的临时凭证进行了身份验证的 API 调用来访问该存储桶。分析师同样尝试使用该角色，但会失败。

此工作流程具有三个基本步骤：

在目标账户中创建角色

首先，您使用 Amazon Web Services 管理控制台 在目标账户（ID 号 999999999999）和原始账户（ID 号 111111111111）之间建立信任。可通过创建名为 UpdateData 的 IAM 角色 角色开始。当您创建角色时，将原始账户定义为受信任实体，并指定一个权限策略，允许受信任用户更新 amzn-s3-demo-bucket-shared-container 存储桶。

向角色授予访问权限

在本节中，您将修改角色策略，以拒绝分析师访问 UpdateData 角色。这种情况下，因为分析师有 PowerUser 访问权限，您必须显式拒绝使用该角色的能力。

通过切换角色测试访问权限

最后，以开发人员的身份使用 UpdateData 角色更新目标账户中的 amzn-s3-demo-bucket-shared-container 存储桶。您可以了解如何通过 Amazon 控制台、Amazon CLI 和 API 访问角色。

注意事项

在使用 IAM 角色跨 Amazon Web Services 账户 委派资源访问权限之前，务必考虑以下几点：

- 以 Amazon Web Services 账户根用户 身份登录后无法切换至角色。

- IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，假定您在标准 aws 分区的美国西部（加利福尼亚北部）中有一个账户。您在 aws-cn 分区的中国（北京）中也有一个账户。您不能使用中国（北京）的账户中 Amazon S3 基于资源的策略，来允许标准 aws 账户中用户的访问权限。
- 您可以使用 Amazon IAM Identity Center 通过安全断言标记语言（SAML）为外部 Amazon Web Services 账户（Amazon Organizations 之外的账户）单点登录（SSO）提供便利。有关详细信息，请参阅 Integrate external Amazon Web Services 账户 into Amazon IAM Identity Center for central access management with independent billing using SAML 2.0
- 您可以将角色与 Amazon EC2 实例或 Amazon Lambda 函数等 Amazon 资源关联。有关更多信息，请参阅 创建向 Amazon 服务委派权限的角色。
- 如果您想让一个应用程序担任另一个 Amazon Web Services 账户 中的角色，则可以使用 Amazon SDK 跨账户担任角色。有关更多信息，请参阅《Amazon SDKs and Tools Reference Guide》中的 Authentication and access。
- 使用 Amazon Web Services 管理控制台 切换角色仅适用于不需要 `ExternalId` 的账户。例如，假定您将您账户的访问权限授予第三方，并在权限策略的 `Condition` 元素中要求 `ExternalId`。在此情况下，第三方只能通过使用 Amazon API 或命令行工具访问您的账户。第三方不能使用控制台，因为它必须提供 `ExternalId` 值。有关此方案的更多信息，请参阅 Amazon 安全博客中的 访问第三方拥有的 Amazon Web Services 账户 和 How to enable cross account access to the Amazon Web Services 管理控制台。

先决条件

本教程假定您已准备好以下各项：

- 您可以使用的两个独立 Amazon Web Services 账户，一个代表原始账户，一个代表目标账户。
- 原始账户中的用户和角色的创建和配置方式如下：

职位	用户	权限
开发者版	David	两个用户均可以登录和使用原始账户中的 Amazon Web Services 管理控制台。
分析师	Jane	

- 您不需要在目标账户中创建任何用户。
- 在目标账户中创建的 Amazon S3 存储桶。在本教程中，您可以将其称为 `amzn-s3-demo-bucket-shared-container`，但由于 S3 存储桶名称必须全局唯一，您必须使用具有其他名称的存储桶。

在目标账户中创建角色

您可以允许一个 Amazon Web Services 账户 中的用户访问其他 Amazon Web Services 账户 中的资源。在本教程中，我们将通过创建一个角色来实现这一点，该角色定义可以访问它的人员以及它向切换到它的用户授予的权限。

在教程的这一步中，您将在目标账户中创建角色，并将原始账户指定为受信任实体。此外，限制更改角色的权限，只有 `amzn-s3-demo-bucket-shared-container` 存储桶的读写权限。任何人只要获得使用该角色的权限，就能对 `shared-container` 存储桶进行读写。

在创建角色之前，您需要原始 Amazon Web Services 账户 的账户 ID。每个 Amazon Web Services 账户 均拥有向其分配的唯一账户 ID 标识符。

获取原始 Amazon Web Services 账户 ID

1. 以原始账户管理员身份登录 Amazon Web Services 管理控制台 并在 https://console.aws.amazon.com/iam/ 上打开 IAM 控制台。
2. 在 IAM 控制台中，在右上角的导航栏上选择您的用户名。它通常类似于：*`username`*@*`account_ID_number_or_alias`*。

   对于此场景，您可以将账户 ID 111111111111 用于原始账户。但是，如果您在测试环境中使用此方案，则应使用有效的账户 ID。

在目标账户中创建可供原始账户使用的角色

1. 以目标账户管理员身份登录 Amazon Web Services 管理控制台 并打开 IAM 控制台。
2. 创建角色之前，准备好定义角色所需权限的托管策略。您可在以后的步骤中将此策略附加到角色。

   您想要设置针对 `amzn-s3-demo-bucket-shared-container` 存储桶的读写权限。尽管 Amazon 提供了一些 Amazon S3 托管策略，但这些策略并未提供对单个 Amazon S3 存储桶的读写访问权限。您可以创建自己的自定义策略。

   在导航窗格中选择策略，然后选择创建策略。
3. 选择 JSON 选项卡，然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中，并将资源 ARN (`arn:aws:s3:::shared-container`) 替换为与您的 Amazon S3 存储桶对应的真实资源 ARN。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListBucket",
        "s3:GetBucketLocation"
       ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-shared-container"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-shared-container/*"
    }
  ]
}
```

`ListAllMyBuckets` 操作会授予列出已通过身份验证的请求发出者所拥有的全部存储桶的权限。`ListBucket` 权限允许用户浏览 `amzn-s3-demo-bucket-shared-container` 存储桶中的对象。`GetObject`、`PutObject`、`DeleteObject` 权限允许用户查看、更新和删除 `amzn-s3-demo-bucket-shared-container` 存储桶中的内容。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

4. 在查看并创建页面上，键入 **`read-write-app-bucket`** 作为策略名称。查看您的策略授予的权限，然后选择创建策略以保存您的工作。

   新策略会显示在托管策略列表中。

5. 在导航窗格中，选择角色，然后选择创建角色。
6. 选择 Amazon Web Services 账户 角色类型。
7. 对于账户 ID，键入原始账户 ID。

   本教程使用原始账户的示例账户 ID **111111111111**。您应使用有效的账户 ID。如果使用无效的账户 ID，如 **111111111111**，IAM 不会让您创建新角色。

   现在，您不必要求外部 ID 或要求用户拥有多重身份验证 (MFA) 就可以担任该角色。将这些选项保持未选中状态。有关更多信息，请参阅 IAM 中的 Amazon 多重身份验证。
8. 选择 Next: Permissions 设置将与角色关联的权限。
9. 选中您之前创建的策略旁边的复选框。

   > **提示**
   >
   > 对于 Filter，选择 Customer managed 对列表进行筛选，使其只包含您创建的策略。这会隐藏 Amazon 创建的策略，更容易找到您所需要的策略。

   然后选择下一步。
10. （可选）以键值对形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
11. （可选）对于 Description（描述），输入新角色的描述。
12. 检查角色后，选择 Create role。

    `UpdateData` 角色显示在角色列表中。

现在，您必须获取该角色的 Amazon Resource Name (ARN)，这是角色的唯一标识。当您修改原始账户中的开发人员角色时，可指定目标账户中的角色 ARN 以授予或拒绝权限。

获取 UpdateData 的 ARN

1. 在 IAM 控制台的导航窗格中，选择角色。
2. 在角色列表中，选择 `UpdateData` 角色。
3. 在详细信息窗格的 Summary (摘要) 部分中，复制 Role ARN (角色 ARN) 值。

   目标账户的账户 ID 是 999999999999，因此角色 ARN 是 `arn:aws:iam::999999999999:role/UpdateData`。确保为目标账户提供真实的 Amazon Web Services 账户 ID。

此时，您已经在目标账户和原始账户之间建立了信任。您通过在目标账户中创建一个角色来将原始账户标识为受信的主体，从而实现这一点。您还可以定义切换到 `UpdateData` 角色的用户可执行哪些操作。

接下来，修改开发人员角色的权限。

向角色授予访问权限

此时，分析师和开发人员都拥有允许其管理原始账户中数据的权限。使用以下必要步骤添加权限以允许切换到角色。

修改开发人员角色以允许其切换到 UpdateData 角色

1. 以原始账户中的管理员身份登录，打开 IAM 控制台。
2. 选择组，然后选择开发人员。
3. 请选择 Permissions（权限）选项卡，然后选择 Add permissions（添加权限），接着选择 Create inline policy（创建内联策略）。
4. 选择 JSON 选项卡。
5. 添加以下策略语句，允许对目标账户中的 `UpdateData` 角色执行 `AssumeRole` 操作。请确保将 `Resource` 元素中的 *`DESTINATION-ACCOUNT-ID`* 更改为目标账户的实际 Amazon Web Services 账户 ID。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::111122223333:role/UpdateData"
    }
}
```

Allow 效果显式允许开发人员组访问目标账户中的 UpdateData 角色。尝试访问该角色的任何开发人员都会成功。

6. 选择查看策略。
7. 键入名称，例如 **allow-assume-S3-role-in-destination**。
8. 选择创建策略。

在大多数环境中，可能不需要执行以下步骤。但是，如果您使用 PowerUserAccess，则某些组可能已能够切换角色。以下过程介绍如何向分析师组添加 "Deny" 权限，以确保其无法担任角色。如果您的环境中不需要此过程，建议不要添加该权限。"Deny" 权限会让整个权限体系更为复杂、难以管理和理解。仅当您没有更好的选择时才使用 "Deny" 权限。

修改分析师角色以拒绝担任 **UpdateData** 角色的权限

1. 选择角色，然后选择分析师。
2. 请选择 Permissions（权限）选项卡，然后选择 Add permissions（添加权限），接着选择 Create inline policy（创建内联策略）。
3. 选择 JSON 选项卡。
4. 添加以下策略语句以拒绝对 AssumeRole 角色执行的 UpdateData 操作。请确保将 Resource 元素中的 *DESTINATION-ACCOUNT-ID* 更改为目标账户的实际 Amazon Web Services 账户 ID。

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": {
        "Effect": "Deny",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::111122223333:role/UpdateData"
    }
}
```

Deny 效果显式拒绝分析师组访问目标账户中的 UpdateData 角色。任何尝试访问该角色的分析师都会收到一条拒绝访问信息。

5. 选择查看策略。
6. 键入名称，例如 **deny-assume-S3-role-in-destination**。
7. 选择创建策略。

开发人员角色现在有使用目标账户中 UpdateData 角色的应权限。禁止分析师角色使用 UpdateData 角色。

接下来，您将了解开发人员 David 如何能访问目标账户中的 amzn-s3-demo-bucket-shared-container 存储桶。David 可以从 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 访问存储桶。

通过切换角色测试访问权限

完成本教程的前两个步骤后，您将具有一个可以授予对目标账户中资源的访问权限的角色。您还将在原始账户中有一个角色，并且用户可以使用该角色。此步骤讨论如何对从 Amazon Web Services 管理控制台、Amazon CLI 和 Amazon API 切换到该角色进行测试。

要获取有关在使用 IAM 角色时可能遇到的常见问题的帮助，请参阅排查 IAM 角色问题。

切换角色（控制台）

如果 David 需要在 Amazon Web Services 管理控制台 中更新目标账户中的数据，则他可以使用切换角色来进行更新。他指定账户 ID 或别名以及角色名称，他的权限会立即切换为该角色允许的权限。然后，他可以通过控制台使用 amzn-s3-demo-bucket-shared-container 存储桶，但是无法使用目标中的其他任何资源。当 David 使用该角色时，他无法使用其在原始账户中的高级用户权限。这是因为，一次仅一组权限能够生效。

David 可以使用 IAM 提供的两种方法进入 Switch Role（切换角色）页：

- David 收到其管理员提供的一个链接，该链接指向一个预定义的“Switch Role”（切换角色）配置。该链接在 Create role 向导最后一页上或在跨账户角色的 Role Summary 页面上提供给管理员。选择该链接将引导 David 进入已填写 Account ID（账户 ID）和 Role name（角色姓名）字段的 Switch Role（切换角色）页面。David 只需选择 Switch Role（切换角色）。
- 管理员不会通过电子邮件发送该链接，而是发送账户 ID 号和角色名称值。要切换角色，David 必须手动输入这些值。以下步骤对此进行说明。

要代入角色

1. David 使用其常规用户以原始账户登录到 Amazon Web Services 管理控制台。
2. 他们选择管理员通过电子邮件向其发送的链接。该会将 David 跳转至 Switch Role（切换角色）页面，其中已填写了账户 ID 或别名和角色名称信息。

   —或者—

   David 在导航栏上选择其姓名 [Identity（身份）菜单]，然后选择 Switch Roles（切换角色）。

   如果这是 David 首次通过这种方式尝试访问“Switch Role (切换角色)”页面，则他会首先登录初始 Switch Role (切换角色) 页面。此页面提供有关切换角色如何使用户可以跨 Amazon Web Services 账户 管理资源的其他信息。David 必须选择此页面上的 Switch Role（切换角色）才能完成此过程的其余步骤。
3. 接下来，为了访问该角色，David 必须手动键入目标账户 ID 号（999999999999）和角色名称（`UpdateData`）。

   此外，David 还想监控当前 IAM 中处于活动状态的角色及相关权限。为了跟踪此信息，他在 Display Name（显示名称）文本框中键入 `Destination`，并选择红色选项，然后选择 Switch Role（切换角色）。
4. David 现在可以通过 Amazon S3 控制台使用 Amazon S3 存储桶，或使用 `UpdateData` 角色有权限的其他任何资源。
5. 完成后，David 可返回其原始权限。为此，他们可以选择导航栏上的目标角色显示名称，然后选择返回 David @ 111111111111。
6. 当 David 下次要切换角色并在导航栏中选择身份菜单时，他将看到目标条目仍在上次的位置。他可以选择该条目立即切换角色，无需重新输入账户 ID 和角色名称。

切换角色 (Amazon CLI)

如果 David 需要在目标环境中的命令行上工作，他可以使用 Amazon CLI 做到这一点。他运行 `aws sts assume-role` 命令并传递角色 ARN 以获取该角色的临时安全凭证。然后，他在环境变量中配置这些凭证，使后续的 Amazon CLI 命令能够使用该角色的权限执行。当 David 使用该角色时，他不能同时使用他在原始账户中的高级用户权限，因为同一时间只有一组权限有效。

请注意，所有访问密钥和令牌都只是示例，不能原样照用。请用您的实际环境的适当值替换。

要代入角色

1. David 打开命令提示符窗口，运行以下命令确认 Amazon CLI 客户端正在工作：

   ```
   aws help
   ```

   > **Note**
   >
   > David 的默认环境使用他通过 `David` 命令创建的默认配置文件中的 `aws configure` 用户凭证。有关更多信息，请参阅《Amazon Command Line Interface 用户指南》中的配置 Amazon Command Line Interface。

2. 他通过运行以下命令开始角色切换过程，以切换到目标账户中的 `UpdateData` 角色。他从创建该角色的管理员处获得了角色 ARN。该命令还需要您提供一个会话名称，您可以选择任何文本作为该名称。

   ```
   aws sts assume-role --role-arn "arn:aws:iam::999999999999:role/UpdateData" --role-
   session-name "David-ProdUpdate"
   ```

   然后 David 在输出中看到以下内容：

   ```
   {
       "Credentials": {
           "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
           "SessionToken": "AQoDYXdzEGcaEXAMPLE2gsYULo
   +Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLE
   CvSRyh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
   uZEXAMPLECihzFB5lTYLto9dyBgSDy
   EXAMPLE9/
   g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3Uuysg
   ```

```
sKdEXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLEsnf87e
NhyDHq6ikBQ==",
        "Expiration": "2014-12-11T23:08:07Z",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

3. David 在输出的“Credentials”（凭证）部分中看到了他们所需要的三个部分。

   - `AccessKeyId`
   - `SecretAccessKey`
   - `SessionToken`

   David 需要配置 Amazon CLI 环境，以在后续的调用中使用这些参数。有关各种凭证配置方法的信息，请参阅配置 Amazon Command Line Interface。您不能使用 `aws configure` 命令，因为它不支持捕获会话令牌。但是，您可手动将信息输入到配置文件中。由于这些是到期时间相对较短的临时凭证，将它们添加到您当前的命令行会话环境中是最简单的。

4. 为了将三个值添加到环境，David 将上一步的输出剪切并粘贴到以下命令中。您可能希望剪切并粘贴到简单文本编辑器中，以解决会话令牌输出中的换行问题。即使此处为清晰起见，将输出显示为换行格式，在添加时也必须是单个长字符串形式。

   以下示例显示了 Windows 环境中的命令，其中“set”是创建环境变量的命令。在 Linux 或 MacOS 计算机上，您将改用“export”命令。该示例的其余部分对于三种环境均有效。

   有关使用适用于 Windows Powershell 的工具的详细信息，请参阅 切换到 IAM 角色（Tools for Windows PowerShell）

```
set AWS_ACCESS_KEY_ID=AKIAIOSFODNN7EXAMPLE
set AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
set AWS_SESSION_TOKEN=AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvS
Ryh0FW7jEXAMPLEW+vE/7s1HRpXviG7b+qYf4nD00EXAMPLEmj4wxS04L/
uZEXAMPLECihzFB5lTYLto9dyBgSDyEXA
MPLEKEY9/
g7QRUhZp4bqbEXAMPLENwGPyOj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UusKd
EXAMPLE1TVastU1A0SKFEXAMPLEiywCC/Cs8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP
+4eZScEXAMPLENhykxiHen
DHq6ikBQ==
```

此时，以下所有命令都在这些凭证确定的角色的权限下运行。对 David 而言就是 UpdateData 角色。

> **Important**
>
> 您可以将常用的配置设置和凭证保存在由 Amazon CLI 维护的文件中。有关更多信息，请参阅《Amazon Command Line Interface User Guide》中的 Using existing configuration and credentials files。

5. 运行该命令访问目标账户中的资源。在此示例中，David 使用以下命令列出其 S3 存储桶的内容。

```
aws s3 ls s3://shared-container
```

因为 Amazon S3 存储桶名称通常是唯一的，所以无需指定拥有存储桶的账户 ID。要访问其他 Amazon 服务的资源，请参阅该服务的 Amazon CLI 文档，了解引用其资源所需使用的命令和语法。

使用 AssumeRole (Amazon API)

当 David 需要通过代码来更新目标账户时，他执行 AssumeRole 调用来担任 UpdateData 角色。该调用返回临时凭证，他可以使用这些凭证访问目标账户中的 amzn-s3-demo-bucket-shared-container 存储桶。使用这些凭证，David 可以调用 API 以更新 amzn-s3-demo-bucket-shared-container 存储桶。但是，即使他在原始账户中拥有高级用户权限，也无法通过调用 API 来访问目标账户中的任何其他资源。

要代入角色

1. David 把 AssumeRole 视为应用程序的一个段调用。他们必须指定 UpdateData ARN：arn:aws:iam::999999999999:role/UpdateData。

   AssumeRole 调用返回的结果包含临时凭证以及 AccessKeyId 和 SecretAccessKey。它还包括一个 Expiration 时间，该时间指示凭证何时到期（届时您必须请求新的凭证）。当您使用 Amazon SDK 设置角色链时，许多凭证提供者会在凭证过期之前自动刷新凭证。

2. David 使用那些临时证书调用 s3:PutObject 升级 amzn-s3-demo-bucket-shared-container 存储段。他们会将凭证作为 AuthParams 参数传递给 API 调用。由于临时角色凭证只有 amzn-s3-demo-bucket-shared-container 存储桶的读写权限，因此对目标账户的任何其他操作都会被拒绝。

有关代码示例 (使用 Python)，请参阅切换到 IAM 角色（Amazon API）。

其他资源

以下资源可帮助您了解有关本教程中主题的更多信息：

- 有关 IAM 用户的更多信息，请参阅 IAM 身份。
- 有关 Amazon S3 存储桶的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的创建存储桶。
- 要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

摘要

您已经完成跨账户 API 访问教程。您创建了一个角色与另一个账户之间建立信任关系，并规定了受信任实体可以执行哪些操作。然后，您修改了角色策略以控制哪些 IAM 用户可以访问该角色。最终，原始账户的开发人员可以使用临时凭证升级目标账户中的 `amzn-s3-demo-bucket-shared-container` 存储桶。

IAM 教程：创建和附加您的第一个客户托管策略

在本教程中，您将使用 Amazon Web Services 管理控制台 创建一个客户管理型策略，然后将该策略附加到您 Amazon Web Services 账户 中的一个 IAM 用户。您创建的策略允许具有只读权限的 IAM 测试用户直接登录 Amazon Web Services 管理控制台。

此工作流程具有三个基本步骤：

步骤 1：创建策略

默认情况下，IAM 用户没有权限来执行任何操作。未经过您的允许，他们无法访问 Amazon 管理控制台，也无法管理其中的数据。在该步骤中，您创建一个允许任何附加的用户登录到该控制台的客户托管策略。

步骤 2：附加策略

将策略附加到用户时，该用户继承与该策略相关的所有访问权限。在此步骤中，您会将新策略附加到测试用户。

步骤 3：测试用户访问权限

附加策略后，您即可以该用户身份登录并测试策略。

先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可作为 IAM 用户身份登录且具有管理权限的 Amazon Web Services 账户。
- 未分配有如下权限或组成员资格的测试 IAM 用户：

用户名称	组	权限
PolicyUser	<无>	<无>

步骤 1：创建策略

在该步骤中，您创建一个允许任何附加用户（具有 IAM 数据的只读访问权限）登录 Amazon Web Services 管理控制台 的客户托管策略。

为测试用户创建策略

1. 使用您具有管理员权限的账户，通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在内容窗格中，选择创建策略。
4. 选择 JSON 选项，然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [ {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    } ]
```

```
}
```

5. 解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项卡中选择 Review policy（查看策略），IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

6. 在查看并创建页面上，键入 **UsersReadOnlyAccessToIAMConsole** 作为策略名称。查看您的策略授予的权限，然后选择创建策略以保存您的工作。

   将在托管策略列表中显示新策略，并已准备好附加该策略。

步骤 2：附加策略

接下来，您会将刚刚创建的策略附加到测试 IAM 用户。

将策略挂载到测试用户

1. 在 IAM 控制台的导航窗格中，选择 Policies（策略）。
2. 在策略列表顶部的搜索框中，开始键入 **UsersReadOnlyAccessToIAMConsole** 直至您的策略出现。然后，选中列表中 UsersReadOnlyAccessToIAMConsole 旁边的单选按钮。
3. 请选择 Actions（操作）按钮，然后选择 Attach（附加）。
4. 在 IAM 实体中，选择选项以筛选用户。
5. 在搜索框中，开始键入 **PolicyUser** 直至该用户在列表上可见。然后，选中列表中该用户旁边的框。
6. 选择附加策略。

您已将策略挂载到 IAM 测试用户，这意味着现在该用户具有 IAM 控制台的只读访问权限。

步骤 3：测试用户访问权限

对于本教程，我们建议您以各测试用户身份登录来测试访问权限，以便能了解用户可能获得的体验。

使用测试用户登录以测试访问权限

1. 使用您的 PolicyUser 测试用户通过以下网址登录到 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 浏览控制台的各个页面并尝试创建新的用户或组。请注意，PolicyUser 可以显示数据，但无法创建或修改现有 IAM 数据。

相关资源

有关信息，请参阅以下资源：

- 托管策略与内联策略
- 控制 IAM 用户对 Amazon Web Services 管理控制台 的访问

摘要

现在，您已成功完成创建和附加客户托管策略所需的所有步骤。因此，您可以使用测试账户登录 IAM 控制台，以了解用户可能获得的体验。

IAM 教程：根据标签定义访问 Amazon 资源的权限

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性被称为“标签”。您可以将标签附加到 IAM 资源（包括 IAM 实体（用户和角色））以及 Amazon 资源。您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。当您使用标签控制对 Amazon 资源的访问时，可通过对 Amazon 策略进行较少更改来实现团队和资源的增长。ABAC 策略比传统 Amazon 策略更加灵活，后者需要您列出每个单独的资源。有关 ABAC 及其相对于传统策略的优势的更多信息，请参阅使用 ABAC 授权根据属性定义权限。

> **Note**
>
> 您必须为每个会话标签传递一个值。Amazon Security Token Service 不支持多值会话标签。

主题

- 教程概述
- 先决条件

- 步骤 1：创建测试用户
- 步骤 2：创建 ABAC 策略
- 步骤 3：创建角色
- 步骤 4：测试创建密钥
- 步骤 5：测试查看密钥
- 步骤 6：测试可扩展性
- 步骤 7：测试更新和删除密钥
- 摘要
- 相关资源
- IAM 教程：将 SAML 会话标签用于 ABAC

教程概述

本教程说明如何创建一个允许具有主体标签的 IAM 角色 角色访问具有匹配标签的资源的策略并测试该策略。当主体向 Amazon 发出请求时，将根据主体标签和资源标签是否匹配来授予其权限。此策略仅允许用户查看或编辑其作业需要的 Amazon 资源。

场景

假设您是一家名为 Example Corporation 的大公司的开发人员主管，同时也是一名经验丰富的 IAM 管理员。您熟悉 IAM 用户、角色和策略的创建和管理。您希望确保您的开发工程师和质量保证团队成员能够访问其所需的资源。您还需要一个随公司发展而扩展的策略。

您选择使用 Amazon 资源标签和 IAM 角色主体标签来为支持 ABAC 策略的服务实施该策略（从 Amazon Secrets Manager 开始）。要了解哪些服务支持基于标签的授权，请参阅使用 IAM 的 Amazon 服务。要了解可以在策略中使用哪些标记条件键以及每个服务的操作和资源，请参阅 Amazon 服务的操作、资源和条件键。您可以对基于 SAML 的身份提供程序或 Web 身份提供程序进行配置，将会话标签传递给 Amazon。当您的员工希望联合身份到 Amazon 中时，其属性将应用到 Amazon 中所得到的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。要了解 SAML 联合身份中的会话标签与本教程的不同之处，请参阅IAM 教程：将 SAML 会话标签用于 ABAC。

您的工程和质量保证团队成员是 Pegasus 或 Unicorn 项目。您可以选择以下 3 个字符的项目和团队标签值：

- `access-project` = `peg`（对于 Pegasus 项目）

- `access-project` = `uni`（对于 Unicorn 项目）
- `access-team` = `eng`（对于工程团队）
- `access-team` = `qas`（对于质量保证团队）

此外，您选择需要 `cost-center` 成本分配标签以启用自定义 Amazon 账单报告。有关更多信息，请参阅《Amazon 账单与成本管理 用户指南》中的使用成本分配标签。

重要决策摘要

- 员工使用 IAM 用户凭证进行登录，然后代入其团队和项目的 IAM 角色 角色。如果您的公司拥有与自己的身份系统，则您可以设置联合身份验证以允许员工在没有 IAM 用户的情况下代入角色。有关更多信息，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。
- 将向所有角色附加同一策略。根据标签允许或拒绝操作。
- 员工可以创建新的资源，但前提是员工必须将相同的标签附加到应用于其角色的资源。这将确保员工能够在创建资源后进行查看。管理员不再需要使用新资源的 ARN 来更新策略。
- 员工可以读取其团队拥有的资源，无论项目如何。
- 员工可以更新和删除其团队和项目拥有的资源。
- IAM 管理员可以为新项目添加新角色。他们可以创建和标记新的 IAM 用户以允许访问相应的角色。管理员不需要编辑策略来支持新项目或团队成员。

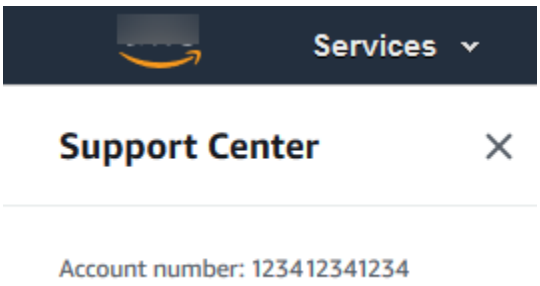
在本教程中，您将标记每个资源，标记项目角色，并将策略添加到角色以允许前面所述的行为。生成的策略允许角色对使用同一项目和团队标签标记的资源进行 `Create`、`Read`、`Update` 和 `Delete` 访问。此策略还允许对使用同一团队标记的资源进行跨项目 `Read` 访问。
此图显示了两个项目，在这两个项目中，角色只能对其项目外的资源进行只读访问，而在其自己的项目中则拥有创建、读取、更新和删除资源的权限。

先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可使用用户身份登录的具有管理权限的 Amazon Web Services 账户。
- 您的 12 位账户 ID（用于在步骤 3 中创建角色）。

  要使用 Amazon Web Services 管理控制台查找 Amazon 账户 ID 号，请在右上角的导航栏上选择 Support (支持)，然后选择 Support Center (支持中心)。账户号 (ID) 将显示在左侧的导航窗格中。



- 体验在 Amazon Web Services 管理控制台 中创建和编辑 IAM 用户、角色和策略。但是，如果您需要获得帮助以便记住 IAM 管理过程，请访问本教程提供的链接来查看分步说明。

步骤 1：创建测试用户

为了进行测试，请创建四个有权代入具有相同标签的角色的 IAM 用户。这可让您更轻松地向您的团队添加更多用户。在标记用户时，他们会自动获得访问权来代入正确角色。如果用户仅在一个项目和团队中工作，则无需将其添加到角色的信任策略中。

1. 创建以下名为 `access-assume-role` 的客户托管策略。有关创建 JSON 策略的更多信息，请参阅创建 IAM 策略。

   ABAC 策略：代入任何 ABAC 角色，但前提是用户和角色标签匹配

   以下策略允许用户在您的账户中代入任何带有 `access-` 名称前缀的角色。还必须使用与用户相同的项目、团队和成本中心来标记角色。

   要使用此策略，请将示例策略中的斜体占位符文本替换为您的账户信息。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::111122223333:role/access-*",
            "Condition": {
                "StringEquals": {
```

```
                    "iam:ResourceTag/access-project": "${aws:PrincipalTag/
access-project}",
                    "iam:ResourceTag/access-team": "${aws:PrincipalTag/
access-team}",
                    "iam:ResourceTag/cost-center": "${aws:PrincipalTag/cost-
center}"
                }
            }
        }
    ]
}
```

要将本教程扩展到大量用户，您可以将策略附加到一个组并将每个用户添加到该组。有关更多信息，请参阅创建 IAM 组和编辑 IAM 组中的用户。

2. 创建以下 IAM 用户，附加 `access-assume-role` 权限策略。确保选择向用户提供对 Amazon Web Services 管理控制台 的访问权限，然后添加以下标签。

用户名称	用户标签键	用户标签值
access-Arnav-peg-eng	access-project	peg
	access-team	eng
	cost-center	987654
access-Mary-peg-qas	access-project	peg
	access-team	qas
	cost-center	987654
access-Saanvi-uni-eng	access-project	uni
	access-team	eng
	cost-center	123456
access-Carlos-uni-qas	access-project	uni
	access-team	qas

用户名称	用户标签键	用户标签值
	cost-center	123456

步骤 2：创建 ABAC 策略

创建以下名为 **access-same-project-team** 的策略。您将在以后的步骤中将此策略添加到角色。有关创建 JSON 策略的更多信息，请参阅创建 IAM 策略。

有关可针对本教程调整的其他策略，请参阅以下页面：

- 控制 IAM 主体进行的访问
- Amazon EC2：允许以编程方式和在控制台中启动或停止用户已标记的 EC2 实例
- EC2：基于匹配的主体和资源标签来启动或停止实例
- EC2：基于标签来启动或停止实例
- IAM：代入具有特定标签的角色

ABAC 策略：仅在主体标签和资源标签匹配时访问 Secrets Manager 资源

以下策略允许主体创建、读取、编辑和删除资源，但前提是已使用与主体相同的键/值对标记这些资源。当主体创建资源时，必须添加 access-project、access-team 和 cost-center 标签以及与主体的标签匹配的值。该策略还允许添加可选的 Name 或 OwnedBy 标签。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
     {
         "Sid": "AllActionsSecretsManagerSameProjectSameTeam",
         "Effect": "Allow",
         "Action": "secretsmanager:*",
         "Resource": "*",
         "Condition": {
             "StringEquals": {
                 "aws:ResourceTag/access-project": "${aws:PrincipalTag/access-
project}",
```

```
                "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-
team}",
                "aws:ResourceTag/cost-center": "${aws:PrincipalTag/cost-center}"
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "access-project",
                    "access-team",
                    "cost-center",
                    "Name",
                    "OwnedBy"
                ]
            },
            "StringEqualsIfExists": {
                "aws:RequestTag/access-project": "${aws:PrincipalTag/access-
project}",
                "aws:RequestTag/access-team": "${aws:PrincipalTag/access-team}",
                "aws:RequestTag/cost-center": "${aws:PrincipalTag/cost-center}"
            }
        }
    },
    {
        "Sid": "AllResourcesSecretsManagerNoTags",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:GetRandomPassword",
            "secretsmanager:ListSecrets"
        ],
        "Resource": "*"
    },
    {
        "Sid": "ReadSecretsManagerSameTeam",
        "Effect": "Allow",
        "Action": [
            "secretsmanager:Describe*",
            "secretsmanager:Get*",
            "secretsmanager:List*"
        ],
        "Resource": "*",
        "Condition": {
            "StringEquals": {
                "aws:ResourceTag/access-team": "${aws:PrincipalTag/access-team}"
            }
        }
```

```
        },
        {
            "Sid": "DenyUntagSecretsManagerReservedTags",
            "Effect": "Deny",
            "Action": "secretsmanager:UntagResource",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringLike": {
                    "aws:TagKeys": "access-*"
                }
            }
        },
        {
            "Sid": "DenyPermissionsManagement",
            "Effect": "Deny",
            "Action": "secretsmanager:*Policy",
            "Resource": "*"
        }
    ]
}
```

此策略有何作用？

- `AllActionsSecretsManagerSameProjectSameTeam` 语句允许对所有相关资源执行此服务的所有操作，但前提是资源标签与主体标签匹配。通过将 `"Action": "secretsmanager:*"` 添加到策略，策略将在 Secrets Manager 增长时扩展。如果 Secrets Manager 添加一个新的 API 操作，则您无需将该操作添加到语句。该语句使用三个条件块实施 ABAC。仅在所有三个块都返回 true 时允许该请求。
  - 如果资源上存在指定的标签键，并且其值与主体的标签匹配，则此语句的第一个条件块将返回 true。对于不匹配的标签或不支持资源标记的操作，此块返回 false。要了解此块不允许哪些操作，请参阅 Amazon Secrets Manager 的操作、资源和条件键。此页面指明对 Secret (密钥) 资源类型执行的操作支持 `secretsmanager:ResourceTag/tag-key` 条件键。某些 Secrets Manager 操作不支持该资源类型，包括 `GetRandomPassword` 和 `ListSecrets`。您必须创建额外语句才能允许这些操作。
  - 如果请求中传递的每个标签键都包含在指定列表中，则第二个条件块将返回 true。通过将 `ForAllValues` 和 `StringEquals` 条件运算符结合使用来执行此操作。如果没有传递键或一组键的子集，则条件返回 true。对于不允许在请求中传递标签的 `Get*` 操作，这将允许该操作。如果请求者包含列表中没有的标签键，则条件返回 false。请求中传递的每个标签键均必须与该列表的一个成员匹配。有关更多信息，请参阅 多值上下文键的集合运算符。

  - 在以下情况下，第三个条件块将返回 true：请求支持传递标签；所有三个标签都存在；标签与主体标签值匹配。如果请求不支持传递标签，则此块也返回 true。这要归功于条件运算符中的 `...IfExists`。如果在支持该块的操作期间未传递任何标签，或者标签键和值不匹配，则该块将返回 false。
- `AllResourcesSecretsManagerNoTags` 语句允许 `GetRandomPassword` 和 `ListSecrets` 操作，而第一个语句不允许这两项操作。
- 如果使用与资源相同的 access-team 标签来标记主体，则 `ReadSecretsManagerSameTeam` 语句允许只读操作。无论项目或成本中心标签如何，都允许此操作。
- `DenyUntagSecretsManagerReservedTags` 语句拒绝从 Secrets Manager 中删除带有以“access-”开头的键的标签的请求。这些标签用于控制对资源的访问，因此删除这些标签可能会删除权限。
- `DenyPermissionsManagement` 语句拒绝访问创建、编辑或删除 Secrets Manager 的基于资源的策略。这些策略可用于更改密钥的权限。

> **Important**
>
> 此策略使用一种策略来允许服务的所有操作，但明确拒绝权限更改操作。拒绝操作将覆盖任何其他允许主体执行该操作的策略。这可能会产生意外结果。作为最佳实践，仅在没有允许该操作的情况下才使用显式拒绝。否则，允许各个操作的列表，默认情况下将拒绝不需要的操作。

步骤 3：创建角色

创建以下 IAM 角色并附加您在上一步中创建的 **`access-same-project-team`** 策略。有关创建 IAM 角色的更多信息，请参阅 创建角色，向 IAM 用户授予权限。如果选择使用联合身份而不是 IAM 用户和角色，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。

作业函数	角色名称	角色标签	角色描述
项目 Pegasus 工厂	access-peg-engineering	access-project = peg access-team = eng cost-center = 987654	允许工程师读取所有工程资源以及创建和管理 Pegasus 工程资源。

作业函数	角色名称	角色标签	角色描述
项目 Pegasus 质量保证	access-peg-quality-assurance	access-project = peg access-team = qas cost-center = 987654	允许 QA 团队读取所有 QA 资源以及创建和管理所有 Pegasus QA 资源。
项目 Unicorn 工程	access-uni-engineering	access-project= uni access-team = eng cost-center = 123456	允许工程师读取所有工程资源以及创建和管理 Unicorn 工程资源。
项目 Unicorn 质量保证	access-uni-quality-assurance	access-project = uni access-team = qas cost-center = 123456	允许 QA 团队读取所有 QA 资源以及创建和管理所有 Unicorn QA 资源。

步骤 4：测试创建密钥

附加到角色的权限策略允许员工创建密钥。仅在使用其项目、团队和成本中心标记密钥时允许这样做。通过在 Secrets Manager 中以用户身份登录、代入正确的角色并测试活动，确认您的权限按预期工作。

测试创建带有和不带有所需标签的密钥

1. 在您的主浏览器窗口中，仍以管理员用户身份登录，以便能在 IAM 中查看用户、角色和策略。使用浏览器无痕窗口或单独的浏览器来进行测试。在那里，以 access-Arnav-peg-eng IAM 用户的身份登录并打开 Secrets Manager 控制台，地址：https://console.aws.amazon.com/secretsmanager/。
2. 尝试切换到 access-uni-engineering 角色。

   此操作失败，因为 access-project 和 cost-center 标签值与 access-Arnav-peg-eng 用户和 access-uni-engineering 角色的不匹配。

有关在 Amazon Web Services 管理控制台 中切换角色的更多信息，请参阅 从用户切换到 IAM 角色（控制台）。

3. 切换到 `access-peg-engineering` 角色。
4. 使用以下信息存储新密钥。要了解如何存储密钥，请参阅 Amazon Secrets Manager 用户指南中的创建基本密钥。

> **Important**
>
> Secrets Manager 会显示提醒，告知您没有权限使用与 Secrets Manager 配合使用的其他 Amazon 服务。例如，要创建 Amazon RDS 数据库凭证，您必须有权描述 RDS 实例、RDS 集群和 Amazon Redshift 集群。您可以忽略这些提醒，因为您没有使用本教程中的这些特定 Amazon 服务。

1. 在 Select secret type (选择密钥类型) 部分中，选择 Other type of secrets (其他类型的密钥)。在两个文本框中，输入 `test-access-key` 和 `test-access-secret`。
2. 为 Secret name (密钥名称) 字段输入 `test-access-peg-eng`。
3. 从下表添加不同的标签组合，并查看预期行为。
4. 选择 Store (存储) 以尝试创建密钥。当存储失败时，请返回之前的 Secrets Manager 控制台页面，并使用下表中的下一个标签集。允许使用最后一个标签集，并且将成功创建密钥。

下表显示了 `test-access-peg-eng` 角色的 ABAC 标签组合。

access-project 标签值	**access-team** 标签值	**cost-center** 标签值	其他标签	预期行为
(无)	(无)	(无)	（无）	已拒绝，因为 `access-project` 标签值与角色的 `peg` 值不匹配。
`uni`	`eng`	`987654`	（无）	已拒绝，因为 `access-project` 标签值与角色的 `peg` 值不匹配。
`peg`	`qas`	`987654`	（无）	已拒绝，因为 `access-team` 标签值与角色的 `eng` 值不匹配。

access-project 标签值	**access-team** 标签值	**cost-center** 标签值	其他标签	预期行为
peg	eng	123456	（无）	已拒绝，因为 cost-center 标签值与角色的 987654 值不匹配。
peg	eng	987654	所有者 = Jane	已拒绝，因为策略不允许额外的标签 owner，即使所有三个所需标签都存在且其值与角色的值匹配也是如此。
peg	eng	987654	名字 = Jane	已允许，因为所有三个所需标签都存在且其值与角色的值匹配。此外，允许您包含可选的 Name 标签。

5. 注销并针对以下每个角色和标签值重复此过程的前三个步骤。在此过程的第四步中，测试所选的任何一组缺少的标签、可选标签、不允许的标签和无效的标签值。然后，使用所需的标签创建具有以下标签和名称的密钥。

用户名称	角色名称	密钥名称	密钥标签
access-Mary-peg-qas	access-peg-quality-assurance	test-access-peg-qas	access-project = peg access-team = qas cost-center = 987654
access-Saanvi-uni-eng	access-uni-engineering	test-access-uni-eng	access-project = uni access-team = eng cost-center = 123456
access-Carlos-uni-qas	access-uni-quality-assurance	test-access-uni-qas	access-project = uni access-team = qas

用户名称	角色名称	密钥名称	密钥标签
			cost-center = 123456

步骤 5：测试查看密钥

您附加到每个角色的策略允许员工查看标记有其团队名称的所有密钥，而不管其项目如何。通过在 Secrets Manager 中测试您的角色来确认您的权限按预期工作。

测试查看带有或不带有所需标签的密钥

1. 以下列某个 IAM 用户身份登录：
   - `access-Arnav-peg-eng`
   - `access-Mary-peg-qas`
   - `access-Saanvi-uni-eng`
   - `access-Carlos-uni-qas`
2. 切换到匹配的角色：
   - `access-peg-engineering`
   - `access-peg-quality-assurance`
   - `access-uni-engineering`
   - `access-uni-quality-assurance`

   有关在 Amazon Web Services 管理控制台中切换角色的更多信息，请参阅从用户切换到 IAM 角色（控制台）。
3. 在左侧导航窗格中，选择菜单图标以展开菜单，然后选择 Secrets (密钥)。
4. 无论您当前的角色如何，都应在表中看到所有四个密钥。这是预期情况，因为名为 `access-same-project-team` 的策略允许对所有资源执行 `secretsmanager:ListSecrets` 操作。
5. 选择其中一个密钥的名称。
6. 在密钥的详细信息页面上，您的角色标签将确定您是否能查看页面内容。将您的角色名称与您的密钥名称进行比较。如果它们共享同一团队名称，则 `access-team` 标签匹配。如果它们不匹配，则访问将被拒绝。

下表显示每个角色的 ABAC 密钥查看行为。

角色名称	密钥名称	预期行为
access-peg-engineering	test-access-peg-eng	允许
	test-access-peg-qas	已拒绝
	test-access-uni-eng	允许
	test-access-uni-qas	已拒绝
access-peg-quality-assurance	test-access-peg-eng	已拒绝
	test-access-peg-qas	允许
	test-access-uni-eng	已拒绝
	test-access-uni-qas	允许
access-uni-engineering	test-access-peg-eng	允许
	test-access-peg-qas	已拒绝
	test-access-uni-eng	允许
	test-access-uni-qas	已拒绝
access-uni-quality-assurance	test-access-peg-eng	已拒绝
	test-access-peg-qas	允许
	test-access-uni-eng	已拒绝
	test-access-uni-qas	允许

7. 从页面顶部的痕迹导航中，选择 Secrets (密钥) 以返回密钥列表。使用不同的角色重复此过程中的步骤来测试您是否能查看每个密钥。

步骤 6：测试可扩展性

在基于角色的访问控制 (RBAC) 上使用基于属性的访问控制 (ABAC) 的重要原因是可扩展性。当您的公司向 Amazon 添加新的项目、团队或人员时，您无需更新 ABAC 驱动型策略。例如，假设 Example Company 正在资助一个代号为 Centaur 的新项目。一个名叫 Saanvi Sarkar 的工程师将成为 Centaur 的工程师主管，同时该工程师还参与了 Unicorn 项目。Saanvi 还将审查 Peg 项目的工作。此外，新聘用了几名工程师，其中包括 Nikhil Jayashankar，他仅参与 Centaur 项目。

向 Amazon 添加新项目

1. 以 IAM 管理员登录，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择 Roles（角色），然后添加名为 `access-cen-engineering` 的 IAM 角色。将 **`access-same-project-team`** 权限策略附加到角色并添加以下角色标签：
   - `access-project` = `cen`
   - `access-team` = `eng`
   - `cost-center` = `101010`
3. 在左侧导航窗格中，请选择用户。
4. 添加一个名为 `access-Nikhil-cen-eng` 的新用户，并附加名为 `access-assume-role` 的策略，然后添加以下用户标签。
   - `access-project` = `cen`
   - `access-team` = `eng`
   - `cost-center` = `101010`
5. 使用步骤 4：测试创建密钥和步骤 5：测试查看密钥中的过程。在另一个浏览器窗口中，测试 Nikhil 是否能创建 Centaur 工程密钥并且是否能查看所有工程密钥。
6. 在您以管理员身份登录的主浏览器窗口中，选择用户 `access-Saanvi-uni-eng`。
7. 在 Permissions (权限) 选项卡上，删除 access-assume-role 权限策略。
8. 添加以下名为 `access-assume-specific-roles` 的内联策略。有关将内联策略添加到用户的更多信息，请参阅为用户或角色嵌入内联策略（控制台）。

   ABAC 策略：仅代入特定角色

   此策略允许 Saanvi 代入 Pegasus 或 Centaur 项目的工程角色。由于 IAM 不支持多值标签，因此有必要创建此自定义策略。不能使用 `access-project` = `peg` 和 `access-project` = `cen` 标记

Saanvi 用户。此外，Amazon 授权模型不能同时与这两个值匹配。有关更多信息，请参阅 在 IAM 和 Amazon STS 中进行标记的规则。相反，您必须手动指定她可以代入的两个角色。

要使用此策略，请将示例策略中的斜体占位符文本替换为您的账户信息。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TutorialAssumeSpecificRoles",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": [
                "arn:aws:iam::111122223333:role/access-peg-engineering",
                "arn:aws:iam::111122223333:role/access-cen-engineering"
            ]
        }
    ]
}
```

9. 使用步骤 4：测试创建密钥和步骤 5：测试查看密钥中的过程。在另一个浏览器窗口中，确认 Saanvi 可以代入这两个角色。检查她是否只能根据角色的标签为项目、团队和成本中心创建密钥。此外，确认她能够查看有关工程团队拥有的所有密钥（包括她刚创建的密钥）的详细信息。

步骤 7：测试更新和删除密钥

附加到角色的 `access-same-project-team` 策略允许员工更新和删除任何标记有项目、团队和成本中心的密钥。通过在 Secrets Manager 中测试您的角色来确认您的权限按预期工作。

测试更新和删除带有和不带有所需标签的密钥

1. 以下列某个 IAM 用户身份登录：

   - `access-Arnav-peg-eng`
   - `access-Mary-peg-qas`
   - `access-Saanvi-uni-eng`
   - `access-Carlos-uni-qas`

- access-Nikhil-cen-eng

2. 切换到匹配的角色：

   - access-peg-engineering
   - access-peg-quality-assurance
   - access-uni-engineering
   - access-peg-quality-assurance
   - access-cen-engineering

   有关在 Amazon Web Services 管理控制台中切换角色的更多信息，请参阅从用户切换到 IAM 角色（控制台）。

3. 对于每个角色，请尝试更新密钥描述，然后尝试删除以下密钥。有关更多信息，请参阅 Amazon Secrets Manager 用户指南中的修改密钥和删除和还原密钥。

   下表显示每个角色的 ABAC 密钥更新和删除行为。

角色名称	密钥名称	预期行为
access-peg-engineering	test-access-peg-eng	允许
	test-access-uni-eng	已拒绝
	test-access-uni-qas	已拒绝
access-peg-quality-assurance	test-access-peg-qas	允许
	test-access-uni-eng	已拒绝
access-uni-engineering	test-access-uni-eng	允许
	test-access-uni-qas	已拒绝
access-peg-quality-assurance	test-access-uni-qas	允许

摘要

您现在已成功完成将标签用于基于属性的访问控制 (ABAC) 所需的所有步骤。您已了解如何定义标记策略。您已将该策略应用于主体和资源。您已创建并应用一个为 Secrets Manager 强制实施该策略的策略。您还了解到，在添加新项目和团队成员时，ABAC 可以轻松扩展。因此，您可以使用测试角色登录 IAM 控制台，并体验如何在 Amazon 中将标签用于 ABAC。

> **Note**
>
> 您添加的策略仅允许在特定条件下执行操作。如果您将不同的策略应用于具有更广泛权限的用户或角色，则可能不会将操作限制为需要标记。例如，如果您使用 `AdministratorAccess`Amazon 托管策略向用户授予完全管理权限，则这些策略不会限制该访问。有关在涉及多个策略时如何确定权限的更多信息，请参阅Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。

相关资源

有关信息，请参阅以下资源：

- 使用 ABAC 授权根据属性定义权限
- Amazon 全局条件上下文密钥
- 创建角色，向 IAM 用户授予权限
- Amazon Identity and Access Management 资源的标签
- 使用标签控制对 Amazon 资源的访问
- 从用户切换到 IAM 角色（控制台）
- IAM 教程：将 SAML 会话标签用于 ABAC

要了解如何监控账户中的标签，请参阅使用无服务器工作流程和 Amazon CloudWatch Events 监控 Amazon 资源的标签更改。

IAM 教程：将 SAML 会话标签用于 ABAC

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于属性来定义权限。在 Amazon 中，这些属性称为标签。您可以将标签附加到 IAM 资源（包括 IAM 实体（用户和角色））以及 Amazon 资源。将实体用于向 Amazon 发出请求时，实体成为主体并且包含标签。

您也可以在代入角色或联合身份用户时传递会话标签。然后，您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。当您使用标签控制对 Amazon 资源的访问时，可通过对 Amazon 策略进行较少更改来实现团队和资源的增长。ABAC 策略比传统 Amazon 策略更加灵活，后者需要您列出每个单独的资源。有关 ABAC 及其相对于传统策略的优势的更多信息，请参阅使用 ABAC 授权根据属性定义权限。

如果您的公司使用基于 SAML 的身份提供程序 (IdP) 管理公司用户身份，则可以在 Amazon 中使用 SAML 属性进行细粒度访问控制。属性包括成本中心标识符、用户电子邮件地址、部门分类和项目分配等。当您将这些属性作为会话标签传递时，可以根据这些会话标签控制对 Amazon 的访问权限。

要通过将 SAML 属性传递给会话主体来完成 ABAC 教程，请完成 IAM 教程：根据标签定义访问 Amazon 资源的权限 中的任务以及本主题中介绍的更改。

先决条件

要执行将 SAML 会话标签用于 ABAC 的步骤，您必须满足以下条件：

- 对基于 SAML 的 IdP 的访问权限，您在其中可以创建具有特定属性的测试用户。
- 能够以具有管理员权限的用户身份登录。
- 体验在 Amazon Web Services 管理控制台 中创建和编辑 IAM 用户、角色和策略。但是，如果您需要获得帮助以便记住 IAM 管理过程，请访问 ABAC 教程提供的链接来查看分步说明。
- 在 IAM 中设置基于 SAML 的 IdP 的经验。要查看详细信息以及指向详细 IAM 文档的链接，请参阅使用 AssumeRoleWithSAML 传递会话标签。

步骤 1：创建测试用户

跳过步骤 1：创建测试用户中的说明。由于您在提供商中定义身份，因此无需为员工添加 IAM 用户。

步骤 2：创建 ABAC 策略

按照 步骤 2：创建 ABAC 策略 中的说明在 IAM 中创建指定的托管策略。

步骤 3：创建和配置 SAML 角色

为 SAML 使用 ABAC 教程时，您必须执行额外的步骤来创建角色、配置 SAML IdP 和启用 Amazon Web Services 管理控制台 访问权限。有关更多信息，请参阅 步骤 3：创建角色。

步骤 3A：创建 SAML 角色

创建一个信任 SAML 身份提供程序和在步骤 1 中创建的 test-session-tags 用户的角色。ABAC 教程使用具有不同角色标签的单独角色。由于您从 SAML IdP 传递会话标签，因此只需要一个角色。要了解如何创建基于 SAML 的角色，请参阅创建用于 SAML 2.0 联合身份验证的角色（控制台）。

将角色命名为 access-session-tags。将 access-same-project-team 权限策略附加到角色。编辑角色信任策略来使用以下策略。有关如何编辑角色的信任关系的详细说明，请参阅更新角色信任策略。

以下角色信任策略允许您的 SAML 身份提供程序和 test-session-tags 用户代入该角色。在代入角色时，他们必须传递三个指定的会话标签。sts:TagSession 操作是允许传递会话标签所必需的。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowSamlIdentityAssumeRole",
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRoleWithSAML",
                "sts:TagSession"
            ],
            "Principal": {"Federated":"arn:aws:iam::123456789012:saml-
provider/ExampleCorpProvider"},
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/cost-center": "*",
                    "aws:RequestTag/access-project": "*",
                    "aws:RequestTag/access-team": [
                        "eng",
                        "qas"
                    ]
                },
                "StringEquals": {"SAML:aud": "https://signin.aws.amazon.com/
saml"}
            }
        }
    ]
}
```

AllowSamlIdentityAssumeRole 语句允许工程和质量保证团队的成员在从示例公司 IdP 将身份联合到 Amazon 中时代入此角色。ExampleCorpProvider SAML 提供商在 IAM 中定义。管理员已经设置 SAML 断言以传递三个必需的会话标签。断言可以传递额外的标签，但必须存在这三个标签。身份的属性可以具有 cost-center 和 access-project 标签的任何值。但是，access-team 属性值必须匹配 eng 或 qas，以指示位于工程或质量保证团队中的身份。

步骤 3B：配置 SAML IdP

配置您的 SAML IdP 以将 cost-center、access-project 和 access-team 属性作为会话标签传递。有关更多信息，请参阅 使用 AssumeRoleWithSAML 传递会话标签。

要将这些属性作为会话标签传递，请在 SAML 断言中包含以下元素。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:cost-center">
  <AttributeValue>987654</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-project">
  <AttributeValue>peg</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:access-team">
  <AttributeValue>eng</AttributeValue>
</Attribute>
```

步骤 3C：启用控制台访问

为联合身份 SAML 用户启用控制台访问权限。有关更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。

步骤 4：测试创建密钥

使用 access-session-tags 角色联合身份到 Amazon Web Services 管理控制台中。有关更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。然后按照说明 步骤 4：测试创建密钥 创建密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。有关更多信息，请参阅 步骤 4：测试创建密钥。

步骤 5：测试查看密钥

按照步骤 5：测试查看密钥中的说明查看在上一步中创建的密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。

步骤 6：测试可扩展性

按照步骤 6：测试可扩展性中的说明测试可扩展性。要执行此操作，您可在基于 SAML 的 IdP 中添加具有以下属性的新身份：

- `cost-center = 101010`
- `access-project = cen`
- `access-team = eng`

步骤 7：测试更新和删除密钥

按照步骤 7：测试更新和删除密钥中的说明更新和删除密钥。使用不同的 SAML 身份，该身份的属性与 ABAC 教程中指示的标签匹配。

> **Important**
>
> 删除您创建的所有密钥以避免产生费用。有关 Secrets Manager 中定价的详细信息，请参阅 Amazon Secrets Manager 定价。

摘要

您现在已成功完成使用 SAML 会话标签和资源标签进行权限管理所需的所有步骤。

> **Note**
>
> 您添加的策略仅允许在特定条件下执行操作。如果您将不同的策略应用于具有更广泛权限的用户或角色，则可能不会将操作限制为需要标记。例如，如果您使用 `AdministratorAccess`Amazon 托管策略向用户授予完全管理权限，则这些策略不会限制该访问。有关在涉及多个策略时如何确定权限的更多信息，请参阅Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。

IAM 教程：允许用户管理其凭证和 MFA 设置

您可以允许用户在安全凭证页面上管理自己的多重身份验证（MFA）设备和凭证。您可以使用 Amazon Web Services 管理控制台为用户配置凭证（访问密钥、密码、签名证书和 SSH 公有密钥）、

删除或停用不再需要的凭证，以及启用 MFA 设备。这在用户数量较少时较为实用，不过随着用户数量增加，这一任务很快会变得非常耗时。本教程旨在介绍如何实现这些最佳实践，而不给您的管理员带来负担。

本教程介绍如何允许用户访问 Amazon 服务，不过此操作仅限于用户使用 MFA 登录的情况。如果未使用 MFA 设备登录，则用户无法访问其他服务。

此工作流程具有三个基本步骤。

步骤 1：创建策略以强制 MFA 登录

创建客户管理型策略以禁止除少量 IAM 操作以外的所有操作。这些例外允许用户在安全凭证页面上更改自己的凭证并管理其 MFA 设备。有关访问该页面的更多信息，请参阅IAM 用户如何更改自己的密码（控制台）。

步骤 2：将策略附加到您的测试用户组

创建一个用户组，在使用 MFA 登录时，该组的成员具有所有 Amazon EC2 操作的完全访问权限。要创建此类用户组，请附加名为 `AmazonEC2FullAccess` 的 Amazon 托管策略和在步骤 1 中创建的客户托管策略。

步骤 3：测试您的用户的访问权限

以测试用户身份登录，验证对 Amazon EC2 的访问是否被阻止，直至该用户创建 MFA 设备为止。之后，用户可以使用该设备进行登录。

先决条件

要执行本教程中的步骤，您必须已具备以下内容：

- 可作为 IAM 用户身份登录且具有管理权限的 Amazon Web Services 账户。
- 您在步骤 1 的策略中键入的账户 ID 号。

  要查找您的账户 ID 号，请在页面顶部的导航栏上，选择 Support，然后选择 Support Center。您可以在该页面的 Support 菜单下找到您的账户 ID。
- 虚拟（基于软件的）MFA 设备、FIDO 安全密钥或基于硬件的 MFA 设备。
- 一个作为组成员的测试 IAM 用户，如下所示：

用户名称	用户名说明	用户组名称	将用户添加为成员	用户组说明
MFAUser	仅选择启用控制台访问 – 可选选项，然后分配密码。	EC2MFA	MFAUser	请勿向此用户组附加任何策略或授予任何权限。

步骤 1：创建策略以强制 MFA 登录

首先创建一个 IAM 客户托管策略，除 IAM 用户管理其自有凭证和 MFA 设备所需的权限外，该策略拒绝其他所有权限。

1. 以具有管理员凭证的用户身份登录 Amazon 管理控制台。根据 IAM 最佳实践，请勿使用 Amazon Web Services 账户根用户凭证登录。

   > **Important**
   >
   > IAM 最佳实践建议您您要求人类用户使用与身份提供商的联合身份验证才能使用临时凭证访问 Amazon，而不是使用具有长期凭证的 IAM 用户。建议您仅在联合用户不支持的特定用例中使用 IAM 用户。

2. 访问：https://console.aws.amazon.com/iam/，打开 IAM 控制台。
3. 在导航窗格中，选择 Policies (策略)，然后选择 Create policy (创建策略)。
4. 选择 JSON 选项卡，然后复制以下 JSON 策略文档中的文本：Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。
5. 将该策略粘贴到 JSON 文本框中。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

   > **Note**
   >
   > 您可以随时在可视化编辑器和 JSON 选项之间切换。但是，以上策略包含 `NotAction` 元素，该元素在可视化编辑器中不受支持。对于此策略，您在 Visual editor 选项卡上将看到一个通知。返回 JSON 以继续处理此策略。
   >
   > 此示例策略不允许用户在首次登录 Amazon Web Services 管理控制台 的同时重置密码。我们建议您在新用户登录并重置密码之前不要向他们授予权限。

6. 在查看并创建页面上，键入 **Force_MFA** 作为策略名称。对于策略描述，在标签区域键入 **This policy allows users to manage their own passwords and MFA devices but nothing else unless they authenticate with MFA.**，您可以有选择地将标签键值对添加到客户管理型策略。查看您的策略授予的权限，然后选择创建策略以保存您的工作。

   将在托管策略列表中显示新策略，并已准备好附加该策略。

步骤 2：将策略附加到您的测试用户组

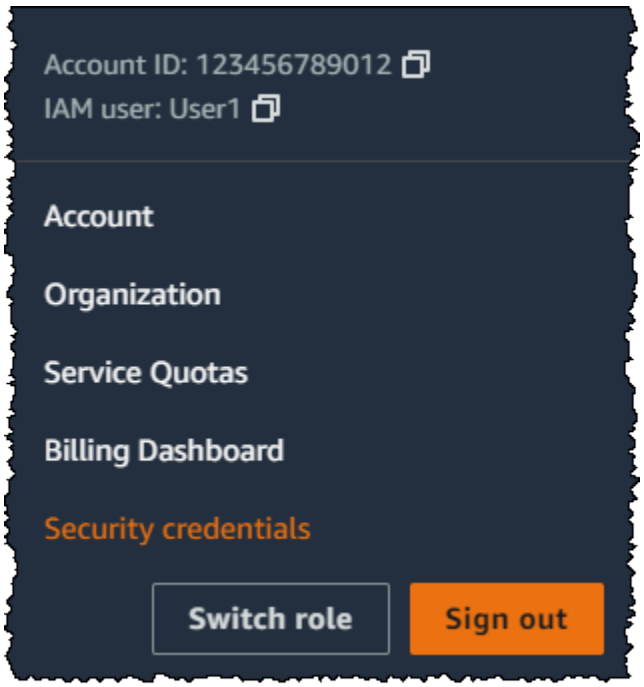
接下来，您将两个策略附加到测试 IAM 用户组，将使用该组授予受 MFA 保护的权限。

1. 在导航窗格中，选择 User groups（用户组）。
2. 在搜索框中键入 **EC2MFA**，然后在列表中选择组名称（而不是复选框）。
3. 请选择权限选项卡，选择添加权限，然后选择附加策略。
4. 在 Attach permission policies to EC2MFA group（将权限策略附加到 EC2MFA 组）页面上，在搜索框中键入 **EC2Full**。然后，选中列表中 AmazonEC2FullAccess 旁边的复选框。暂不保存您的更改。
5. 在搜索框中键入 **Force**，然后选中列表中 Force_MFA 旁边的复选框。
6. 选择附加策略。

步骤 3：测试您的用户的访问权限

在本教程的这一部分中，您以测试用户身份登录并验证策略是否正常工作。

1. 使用上一部分中分配给您的密码，以 **MFAUser** 身份登录您的 Amazon Web Services 账户。使用 URL：`https://<alias or account ID number>.signin.aws.amazon.com/console`
2. 选择 EC2 打开 Amazon EC2 控制台，验证此用户没有执行任何操作的权限。
3. 在右上角的导航栏中，选择 `MFAUser` 用户名，然后选择 Security credentials（安全凭证）。



4. 现在添加一个 MFA 设备。在多重身份验证 (MFA) 部分中，选择分配 MFA 设备。

> **Note**
>
> 可能会出现未授权您执行 `iam:DeleteVirtualMFADevice` 的错误。如果某个人以前开始将虚拟 MFA 设备分配给该用户并取消了该过程，则可能会发生这种情况。要继续，您或其他管理员必须删除用户现有的未分配虚拟 MFA 设备。有关更多信息，请参阅 我没有权限执行：iam:DeleteVirtualMFADevice。

5. 对于本教程，我们使用一个虚拟 (基于软件的) MFA 设备，例如，手机上的 Google Authenticator 应用程序。选择 Authenticator app（身份验证器应用程序），然后单击 Next（下一步）。

   IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。

6. 打开您的虚拟 MFA 应用程序。（有关可以用作托管虚拟 MFA 设备的应用程序的列表，请参阅虚拟 MFA 应用程序。）如果虚拟 MFA 应用程序支持多个账户 (多个虚拟 MFA 设备)，请选择相应的选项以创建新账户 (新的虚拟 MFA 设备)。

7. 确定 MFA 应用程序是否支持 QR 代码，然后执行以下操作之一：

   - 在向导中，选择 Show QR code (显示 QR 代码)。使用此应用程序扫描 QR 代码。例如，您可选择摄像头图标或选择类似于 Scan code 的选项，然后使用设备的摄像头扫描此代码。

- 在 Set up device（设置设备）向导中，选择 Show secret key（显示私有密钥），然后在您的 MFA 应用程序中键入私有密钥。

完成操作后，虚拟 MFA 设备会开始生成一次性密码。

8. 在 Set up device（设置设备）向导的 Enter the code from your authenticator app.（键入身份验证器应用程序中的代码。）框中，键入虚拟 MFA 设备上当前显示的一次性密码。选择 Register MFA（注册 MFA）。

> **Important**
>
> 生成代码之后立即提交您的请求。如果在生成代码后等待很长时间才提交请求，MFA 设备将成功与用户关联，但 MFA 设备不同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

虚拟 MFA 设备现已准备好与 Amazon 一起使用。

9. 从控制台注销，然后重新以 **`MFAUser`** 身份登录。此时 Amazon 会提示输入您手机中的 MFA 代码。收到该代码后，请将代码键入框中，然后选择 Submit。
10. 选择 EC2 以再次打开 Amazon EC2 控制台。请注意，这次您可以看到所有信息，并且可以执行所需的任何操作。如果您以该用户身份转至任何其他控制台，则会看到指示访问被拒绝的消息。导致出现此问题的原因是，本教程中的策略仅向 Amazon EC2 授予访问权限。

相关资源

有关更多信息，请参阅以下主题：

- IAM 中的 Amazon 多重身份验证
- 已启用 MFA 的登录

IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者 (IdP)

要为您的 Amazon 账户设置 SAML 联合身份验证，需要创建 SAML 身份提供者 (IdP)。本教程展示如何使用 Amazon CloudFormation 模板创建 SAML IdP，以便在 Amazon 和外部 IdP 之间建立信任。

该模板会创建一个使用 IdP 元数据文档进行配置的 SAML IdP。然后，联合 IAM 角色可引用此 IdP，以允许来自外部 IdP 的经过身份验证的用户访问 Amazon 资源。

所部署的资源中含有一个配置了 IdP 元数据文档和可选加密设置的 SAML IdP。

先决条件

本教程假定您已准备好以下各项：

- 本地计算机上安装了 Python 3.6 或更高版本，可运行本教程中用于对 IdP 的 SAML 元数据 XML 文件进行格式化的 Python 命令。
- 来自外部 IdP 的 SAML 元数据文档，另存为 XML 文件。

使用 Amazon CloudFormation 创建 SAML IdP

要创建 SAML IdP，您需要创建 CloudFormation 模板，并使用此模板创建包含 IdP 资源的堆栈。

创建模板

首先，创建 CloudFormation 模板。

1. 在 模板 部分中，单击 JSON 或 YAML 选项卡上的复制图标，以复制模板内容。
2. 将模板内容粘贴到新文件中。
3. 将该文件保存在本地。

创建堆栈

接下来，使用已保存的模板来预置 CloudFormation 堆栈。

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面，从创建堆栈菜单中选择（采用新资源（标准））。
3. 指定模板：
   a. 在先决条件下，选择选择现有模板。
   b. 在指定模板下，选择上传模板文件。
   c. 选择选择文件，导航到模板文件并将其选中。

d. 选择下一步。

4. 指定以下堆栈详细信息：

   a. 输入堆栈名称。

   b. 对于 IdentityProviderName，可将其留空以根据堆栈名称自动生成名称，也可以为 SAML IdP 输入自定义名称。自定义名称只能包含字母数字字符、句点、下划线和连字符。

   c. 对于 IdentityProvidersamlMetadataDocument，在将 SAML 元数据 XML 文件粘贴到此字段之前，需要将其格式化为单行。这是必要的，因为 CloudFormation 控制台要求 XML 内容在通过控制台参数传递时需格式化为单行。

      使用以下 Python 命令重新格式化 XML 文件：

```
python3 -c "import sys, re; content=open(sys.argv[1]).read(); print(re.sub(r'>
\s+<', '><', content.replace('\n', '').replace('\r', '').strip()))" saml-
metadata.xml
```

> **Note**
>
> IdP 的 SAML 元数据文档必须格式化为单行才能输入控制台参数。Python 命令删除了换行符和多余空格以创建所需的格式，同时保留所有原始内容和结构。

      复制 Python 命令的输出并将其粘贴到 IdentityProviderSAMLMetadataDocument 字段中。

      格式化的 SAML 元数据文档示例（缩减版）：

```
<?xml version="1.0" encoding="UTF-8"?><md:EntityDescriptor
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 entityID="https://portal.sso.example.com/saml/assertion/
CompanyIdP"><md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><md:KeyDescriptor
 use="signing"><ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/
xmldsig#"><ds:X509Data><ds:X509Certificate>MIIDXTCCAkWgAwIBAgIJAJC1HiIAZAiIMA0GCSqGSIb3
ds:X509Certificate></ds:X509Data></ds:KeyInfo></
md:KeyDescriptor><md:SingleLogoutService
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
 Location="https://portal.sso.example.com/saml/logout/
CompanyIdP"/><md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</md:NameIDFormat><md:SingleSignOnService
```

```
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
portal.sso.example.com/saml/assertion/CompanyIdP"/></md:IDPSSODescriptor></
md:EntityDescriptor>
```

d. 对于其他参数，请接受默认值或根据需要输入自己的值：

- IdentityProviderAddPrivateKey - 解密 SAML 断言的可选私钥
- IdentityProviderAssertionEncryptionMode - 可选项，可设置 SAML 断言加密模式（“允许”、“必需”或空）

e. 选择下一步。

5. 配置堆栈选项：

a. 在堆栈故障选项下，选择删除所有新创建的资源。

> **Note**
>
> 如果选择此选项，即使堆栈创建失败，您可能也不会因为删除策略规定保留资源而向资源付费。

b. 接受其他所有默认值。

c. 在功能下勾选复选框，确认 CloudFormation 可能在您的账户中创建 IAM 资源。

d. 选择下一步。

6. 查看堆栈详细信息并选择提交。

Amazon CloudFormation 将创建堆栈。一旦堆栈创建完成之后，堆栈资源就即可投入使用。可使用堆栈详细信息页面上的资源选项卡来查看账户中预置的资源。

堆栈将输出以下值，可参见输出选项卡：

- ProviderARN：已创建的 SAML IdP 的 ARN（例如，`arn:aws:iam::123456789012:saml-provider/CompanyIdP`）。在创建信任此提供商的角色时需要此 ARN。
- ProviderName：已创建的 SAML IdP 的名称（例如 `CompanyIdP`，如果指定自定义名称；或者 `my-saml-stack-saml-provider`，如果使用默认命名）。

这些输出还会被导出，以便由其他 Amazon CloudFormation 堆栈使用 `Fn::ImportValue` 函数进行导入。

验证 SAML IdP

一旦创建 SAML IdP 后，即可验证其配置并记下其 ARN 以供联合角色使用。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。

   您应该能在列表中看到新创建的 SAML IdP。
3. 选择 IdP 名称以查看其详细信息。

   在 IdP 详细信息页面上，可以看到 SAML 元数据文档和其他配置详细信息。
4. 记下详细信息页面上显示的提供者 ARN。

   在创建信任此 IdP 的联合 IAM 角色时需要此 ARN。
5. 查看元数据文档，确保其与您从外部 IdP 提供的文档相符。

您的 SAML IdP 现已准备好供联合 IAM 角色使用。您可以创建信任此 IdP 的角色，以允许来自外部 IdP 的经过身份验证的用户代入这些角色并访问 Amazon 资源。

清理：删除资源

最后一步，您将删除该堆栈及其包含的资源。

1. 打开 Amazon CloudFormation 控制台。
2. 在堆栈页面上，选择根据该模板创建的堆栈，选择删除，然后确认删除。

   CloudFormation 启动删除堆栈及其包含的所有资源。

CloudFormation 模板详细信息

资源

本教程的 Amazon CloudFormation 模板将在您的账户中创建以下资源：

`AWS::IAM::SAMLProvider`：一个在 Amazon 和外部 IdP 之间建立信任的 SAML IdP。

配置

模板包含如下可配置参数：

- IdentityProviderName - SAML IdP 的名称（自动生成的名字请留空）

  示例：`CompanyIdP` 或 `EnterpriseSSO`
- IdentityProviderSAMLMetadataDocument - 来自外部 IdP 的 SAML 元数据文档（格式化为单行）
- IdentityProviderAddPrivateKey - 解密 SAML 断言的可选私钥
- IdentityProviderAssertionEncryptionMode - 可选项，可设置 SAML 断言加密模式

CloudFormation 模板

将以下 JSON 或 YAML 代码另存为单独文件，以用作本教程的 CloudFormation 模板。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "[AWSDocs] IAM: tutorial_saml-idp",
  "Parameters": {
    "IdentityProviderName": {
      "Type": "String",
      "Description": "Name of the SAML Identity Provider (leave empty for auto-
generated name like '{StackName}-{UniqueId}')",
      "Default": "",
      "AllowedPattern": "^$|^[a-zA-Z0-9._-]+$",
      "ConstraintDescription": "Must be empty or contain only alphanumeric
 characters, periods, underscores, and hyphens"
    },
    "IdentityProviderSAMLMetadataDocument": {
      "Type": "String",
      "Description": "SAML metadata document from identity provider"
    },
    "IdentityProviderAddPrivateKey": {
      "Type": "String",
      "Description": "Optional private key for decrypting SAML assertions. The
 private key must be a .pem file that uses AES-GCM or AES-CBC encryption algorithm
 to decrypt SAML assertions.",
      "Default": ""
    },
    "IdentityProviderAssertionEncryptionMode": {
      "Type": "String",
      "Description": "Optional, sets encryption mode for SAML assertions",
      "Default": "",
```

```
      "AllowedValues": ["", "Allowed", "Required"]
    }
  },
  "Conditions": {
    "HasPrivateKey": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"IdentityProviderAddPrivateKey"}, ""]}]},
    "HasEncryptionMode": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"IdentityProviderAssertionEncryptionMode"}, ""]}]},
    "HasCustomName": {"Fn::Not": [{"Fn::Equals": [{"Ref": "IdentityProviderName"},
""]}]}
  },
  "Resources": {
    "SAMLProvider": {
      "Type": "AWS::IAM::SAMLProvider",
      "Properties": {
        "Name": {"Fn::If": ["HasCustomName", {"Ref": "IdentityProviderName"},
{"Ref": "AWS::NoValue"}]},
        "SamlMetadataDocument": {"Ref": "IdentityProviderSAMLMetadataDocument"},
        "Tags": [
          {
            "Key": "Name",
            "Value": {"Fn::If": ["HasCustomName", {"Ref": "IdentityProviderName"},
{"Fn::Sub": "${AWS::StackName}-saml-provider"}]}
          }
        ],
        "AddPrivateKey": {"Fn::If": ["HasPrivateKey", {"Ref":
"IdentityProviderAddPrivateKey"}, {"Ref": "AWS::NoValue"}]},
        "AssertionEncryptionMode": {"Fn::If": ["HasEncryptionMode", {"Ref":
"IdentityProviderAssertionEncryptionMode"}, {"Ref": "AWS::NoValue"}]}
      }
    }
  },
  "Outputs": {
    "ProviderARN": {
      "Description": "ARN of the created SAML Identity Provider",
      "Value": {"Ref": "SAMLProvider"},
      "Export": {
        "Name": {"Fn::Sub": "${AWS::StackName}-ProviderARN"}
      }
    },
    "ProviderName": {
      "Description": "Name of the SAML Identity Provider",
      "Value": {"Fn::If": ["HasCustomName", {"Ref": "IdentityProviderName"},
{"Fn::Sub": "${AWS::StackName}-saml-provider"}]},
```

```
      "Export": {
        "Name": {"Fn::Sub": "${AWS::StackName}-ProviderName"}
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWSDocs] IAM: tutorial_saml-idp'

Parameters:
  IdentityProviderName:
    Type: String
    Description: Name of the SAML Identity Provider (leave empty for auto-generated
 name like '{StackName}-{UniqueId}')
    Default: ""
    AllowedPattern: '^$|^[a-zA-Z0-9._-]+$'
    ConstraintDescription: 'Must be empty or contain only alphanumeric characters,
 periods, underscores, and hyphens'

  IdentityProviderSAMLMetadataDocument:
    Type: String
    Description: SAML metadata document from identity provider

  IdentityProviderAddPrivateKey:
    Type: String
    Description: Optional private key for decrypting SAML assertions. The private
 key must be a .pem file that uses AES-GCM or AES-CBC encryption algorithm to
 decrypt SAML assertions.
    Default: ""

  IdentityProviderAssertionEncryptionMode:
    Type: String
    Description: Optional, sets encryption mode for SAML assertions
    Default: ""
    AllowedValues:
      - ""
      - "Allowed"
      - "Required"

Conditions:
```

```
  HasPrivateKey: !Not [!Equals [!Ref IdentityProviderAddPrivateKey, ""]]
  HasEncryptionMode: !Not [!Equals [!Ref IdentityProviderAssertionEncryptionMode,
 ""]]
  HasCustomName: !Not [!Equals [!Ref IdentityProviderName, ""]]

Resources:
  SAMLProvider:
    Type: 'AWS::IAM::SAMLProvider'
    Properties:
      Name: !If
        - HasCustomName
        - !Ref IdentityProviderName
        - !Ref AWS::NoValue
      SamlMetadataDocument: !Ref IdentityProviderSAMLMetadataDocument
      Tags:
        - Key: Name
          Value: !If
            - HasCustomName
            - !Ref IdentityProviderName
            - !Sub '${AWS::StackName}-saml-provider'
      AddPrivateKey: !If
        - HasPrivateKey
        - !Ref IdentityProviderAddPrivateKey
        - !Ref AWS::NoValue
      AssertionEncryptionMode: !If
        - HasEncryptionMode
        - !Ref IdentityProviderAssertionEncryptionMode
        - !Ref AWS::NoValue

Outputs:
  ProviderARN:
    Description: 'ARN of the created SAML Identity Provider'
    Value: !Ref SAMLProvider
    Export:
      Name: !Sub '${AWS::StackName}-ProviderARN'

  ProviderName:
    Description: 'Name of the SAML Identity Provider'
    Value: !If
      - HasCustomName
      - !Ref IdentityProviderName
      - !Sub '${AWS::StackName}-saml-provider'
    Export:
```

```
      Name: !Sub '${AWS::StackName}-ProviderName'
```

IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 联合身份验证 IAM 角色

当 Amazon 账户中配置了现有 SAML 身份提供者 (IdP) 时，可创建信任该 IdP 的联合 IAM 角色。本教程展示如何使用 Amazon CloudFormation 模板，创建通过外部 IdP 进行身份验证的用户可代入的 SAML 联合身份验证 IAM 角色。

该模板使用允许 SAML IdP 代入该角色的信任策略创建联合 IAM 角色。经外部 IdP 进行身份验证的用户可以代入此角色，根据附加至该角色的权限访问 Amazon 资源。

所部署的资源包括以下部分：

- 信任现有 SAML IdP 的联合 IAM 角色。
- 可配置的托管策略，可以附加到该角色以授予特定权限。
- 可选的权限边界和会话持续时间设置。

先决条件

本教程假定您已准备好以下各项：

- Amazon 账户中配置的现有 SAML IdP。如果还没有，可以使用 IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者 (IdP) 教程创建一个。
- SAML IdP 的 ARN，需要在创建堆栈时将其指定为参数。
- 本地计算机上安装了 Python 3.6 或更高版本，可运行本教程中用于对 IdP 的 SAML 元数据 XML 文件进行格式化的 Python 命令。

使用 Amazon CloudFormation 创建 SAML 联合角色

要创建 SAML 联合身份验证角色，您需要创建 CloudFormation 模板，并使用此模板创建包含该角色的堆栈。

创建模板

首先，创建 CloudFormation 模板。

1. 在 模板 部分中，单击 JSON 或 YAML 选项卡上的复制图标，以复制模板内容。
2. 将模板内容粘贴到新文件中。
3. 将该文件保存在本地。

创建堆栈

接下来，使用已保存的模板来预置 CloudFormation 堆栈。

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面，从创建堆栈菜单中选择（采用新资源（标准））。
3. 指定模板：
   a. 在先决条件下，选择选择现有模板。
   b. 在指定模板下，选择上传模板文件。
   c. 选择选择文件，导航到模板文件并将其选中。
   d. 选择下一步。
4. 指定以下堆栈详细信息：
   a. 输入堆栈名称。
   b. 对于 SAMLProviderARN，请输入现有 SAML IdP 的 ARN。其格式应为 `arn:aws:iam::123456789012:saml-provider/YourProviderName`。

      示例：`arn:aws:iam::123456789012:saml-provider/CompanyIdP`

      > **Note**
      > 如果使用 IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者 (IdP) 教程创建了 SAML IdP，则可以在该 CloudFormation 堆栈的“输出”选项卡中找到提供者 ARN。

   c. 对于 RoleName，可将其留空以根据堆栈名称自动生成名称，也可以为 IAM 角色输入自定义名称。

      示例：`SAML-Developer-Access` 或 `SAML-ReadOnly-Role`

   d. 对于其他参数，请接受默认值或根据需要输入自己的值：

- RoleSessionDuration - 最大会话持续时间，以秒为单位（3600-43200，默认为 7200）

  示例：`14400`（4 小时）

- RolePermissionsBoundary - 权限边界策略的可选 ARN

  示例：`arn:aws:iam::123456789012:policy/DeveloperBoundary`

- RolePath - IAM 角色的路径（默认为 /）

  示例：`/saml-roles/`

- ManagedPolicy1-5 - 最多 5 个待附加托管策略的可选 ARN

  ManagedPolicy1 示例：`arn:aws:iam::aws:policy/ReadOnlyAccess`

  ManagedPolicy2 示例：`arn:aws:iam::123456789012:policy/CustomPolicy`

e. 选择下一步。

5. 配置堆栈选项：

   a. 在堆栈故障选项下，选择删除所有新创建的资源。

   > **Note**
   >
   > 如果选择此选项，即使堆栈创建失败，您可能也不会因为删除策略规定保留资源而向资源付费。

   b. 接受其他所有默认值。

   c. 在功能下勾选复选框，确认 CloudFormation 可能在您的账户中创建 IAM 资源。

   d. 选择下一步。

6. 查看堆栈详细信息并选择提交。

Amazon CloudFormation 将创建堆栈。一旦堆栈创建完成之后，堆栈资源就即可投入使用。可使用堆栈详细信息页面上的资源选项卡来查看账户中预置的资源。

堆栈将输出以下值，可参见输出选项卡：

- RoleARN：已创建的 IAM 角色的 ARN（例如，`arn:aws:iam::123456789012:role/SAML-Developer-Access` 或 `arn:aws:iam::123456789012:role/stack-name-a1b2c3d4`，如果使用自动生成的名称）。

在配置 IdP 以发送相应 SAML 属性进行角色代入时，需要此角色 ARN。

测试 SAML 联合角色

一旦创建 SAML 联合角色后，可验证其配置并测试联合身份验证设置。

1. 通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 找到并选择新创建的联合角色。

   如果提供了自定义角色名称，请查找该名称。如果将 RoleName 参数留空，则该角色将拥有一个基于堆栈名称和唯一标识符的自动生成名称。
4. 选择信任关系选项卡，查看信任策略。

   信任策略应表明在 SAML 受众 (SAML:aud) 匹配 `https://signin.aws.amazon.com/saml` 的条件下，信任您的 SAML IdP 代入此角色。
5. 选择权限选项卡，查看附加的策略。

   您可以看到在创建过程中附加至该角色的任何托管策略。
6. 记下角色摘要页面上显示的角色 ARN。

   您需要此 ARN 来配置外部 IdP，以允许用户代入此角色。

您的 SAML 联合角色已准备好投入使用。将外部 IdP 配置为在 SAML 断言中包含此角色的 ARN，经过身份验证的用户将能够代入此角色来访问 Amazon 资源。

清理：删除资源

最后一步，您将删除该堆栈及其包含的资源。

1. 打开 Amazon CloudFormation 控制台。
2. 在堆栈页面上，选择根据该模板创建的堆栈，选择删除，然后确认删除。

   CloudFormation 启动删除堆栈及其包含的所有资源。

CloudFormation 模板详细信息

资源

本教程的 Amazon CloudFormation 模板将在您的账户中创建以下资源：

- `AWS::IAM::Role`：一个可以由通过 SAML IdP 进行身份验证的用户代入的联合 IAM 角色。

配置

模板包含如下可配置参数：

- RoleName - IAM 角色的名称（自动生成的名称请留空）
- SAMLProviderARN - SAML IdP 的 ARN（必需）
- RoleSessionDuration - 最大会话持续时间，以秒为单位（3600-43200，默认为 7200）
- RolePermissionsBoundary - 权限边界策略的可选 ARN
- RolePath - IAM 角色的路径（默认为 /）
- ManagedPolicy1-5 - 最多 5 个待附加托管策略的可选 ARN

CloudFormation 模板

将以下 JSON 或 YAML 代码另存为单独文件，以用作本教程的 CloudFormation 模板。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "[AWSDocs] IAM: tutorial_saml-federated-role",
  "Parameters": {
    "RoleName": {
      "Type": "String",
      "Description": "Name of the IAM Role (leave empty for auto-generated name like
 '{StackName}-{UniqueId}')",
      "Default": "",
      "AllowedPattern": "^$|^[\\w+=,.@-]{1,64}$",
      "ConstraintDescription": "Must be empty or 1-64 characters and can contain
 alphanumeric characters and +=,.@-"
    },
    "SAMLProviderARN": {
```

```
      "Type": "String",
      "Description": "ARN of the SAML Identity Provider",
      "AllowedPattern": "^arn:aws:iam::\\d{12}:saml-provider/[a-zA-Z0-9._-]+$",
      "ConstraintDescription": "Must be a valid SAML provider ARN"
    },
    "RoleSessionDuration": {
      "Type": "Number",
      "Description": "The maximum session duration (in seconds) that you want to set
for the specified role (3600-43200)",
      "MinValue": 3600,
      "MaxValue": 43200,
      "Default": 7200
    },
    "RolePermissionsBoundary": {
      "Type": "String",
      "Description": "Optional ARN of the permissions boundary policy (leave empty
for none)",
      "Default": ""
    },
    "RolePath": {
      "Type": "String",
      "Description": "Path for the IAM role (must start and end with /)",
      "Default": "/",
      "AllowedPattern": "^\/.*\/$|^\/$",
      "ConstraintDescription": "Role path must start and end with forward slash (/)"
    },
    "RoleManagedPolicy1": {
      "Type": "String",
      "Description": "Optional managed policy ARN 1",
      "Default": ""
    },
    "RoleManagedPolicy2": {
      "Type": "String",
      "Description": "Optional managed policy ARN 2",
      "Default": ""
    },
    "RoleManagedPolicy3": {
      "Type": "String",
      "Description": "Optional managed policy ARN 3",
      "Default": ""
    },
    "RoleManagedPolicy4": {
      "Type": "String",
      "Description": "Optional managed policy ARN 4",
```

```
      "Default": ""
    },
    "RoleManagedPolicy5": {
      "Type": "String",
      "Description": "Optional managed policy ARN 5",
      "Default": ""
    }
  },
  "Conditions": {
    "HasCustomRoleName": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleName"}, ""]}]},
    "HasPermissionsBoundary": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"RolePermissionsBoundary"}, ""]}]},
    "HasPolicy1": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy1"},
""]}]},
    "HasPolicy2": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy2"},
""]}]},
    "HasPolicy3": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy3"},
""]}]},
    "HasPolicy4": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy4"},
""]}]},
    "HasPolicy5": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy5"}, ""]}]}
  },
  "Resources": {
    "SAMLFederatedRole": {
      "Type": "AWS::IAM::Role",
      "Properties": {
        "RoleName": {"Fn::If": ["HasCustomRoleName", {"Ref": "RoleName"}, {"Ref":
"AWS::NoValue"}]},
        "Description": "IAM role with SAML provider trust",
        "MaxSessionDuration": {"Ref": "RoleSessionDuration"},
        "PermissionsBoundary": {"Fn::If": ["HasPermissionsBoundary", {"Ref":
"RolePermissionsBoundary"}, {"Ref": "AWS::NoValue"}]},
        "Path": {"Ref": "RolePath"},
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Federated": {"Ref": "SAMLProviderARN"}
              },
              "Action": "sts:AssumeRoleWithSAML",
              "Condition": {
                "StringEquals": {
```

```
                  "SAML:aud": "https://signin.aws.amazon.com/saml"
                }
              }
            }
          ]
        },
        "ManagedPolicyArns": {
          "Fn::Split": [
            ",",
            {
              "Fn::Join": [
                ",",
                [
                  {"Fn::If": ["HasPolicy1", {"Ref": "RoleManagedPolicy1"}, {"Ref":
 "AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy2", {"Ref": "RoleManagedPolicy2"}, {"Ref":
 "AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy3", {"Ref": "RoleManagedPolicy3"}, {"Ref":
 "AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy4", {"Ref": "RoleManagedPolicy4"}, {"Ref":
 "AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy5", {"Ref": "RoleManagedPolicy5"}, {"Ref":
 "AWS::NoValue"}]}
                ]
              ]
            }
          ]
        }
      }
    }
  },
  "Outputs": {
    "RoleARN": {
      "Description": "ARN of the created IAM Role",
      "Value": {"Fn::GetAtt": ["SAMLFederatedRole", "Arn"]},
      "Export": {
        "Name": {"Fn::Sub": "${AWS::StackName}-RoleARN"}
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWSDocs] IAM: tutorial_saml-federated-role'

Parameters:
  RoleName:
    Type: String
    Description: 'Name of the IAM Role (leave empty for auto-generated name like
 ''{StackName}-{UniqueId}'')'
    Default: ""
    AllowedPattern: '^$|^[\w+=,.@-]{1,64}$'
    ConstraintDescription: 'Must be empty or 1-64 characters and can contain
 alphanumeric characters and +=,.@-'

  SAMLProviderARN:
    Type: String
    Description: 'ARN of the SAML Identity Provider'
    AllowedPattern: '^arn:aws:iam::\d{12}:saml-provider/[a-zA-Z0-9._-]+$'
    ConstraintDescription: 'Must be a valid SAML provider ARN'

  RoleSessionDuration:
    Type: Number
    Description: 'The maximum session duration (in seconds) that you want to set for
 the specified role (3600-43200)'
    MinValue: 3600
    MaxValue: 43200
    Default: 7200

  RolePermissionsBoundary:
    Type: String
    Description: Optional ARN of the permissions boundary policy (leave empty for
 none)
    Default: ""

  RolePath:
    Type: String
    Description: 'Path for the IAM role (must start and end with /)'
    Default: "/"
    AllowedPattern: '^\/.*\/$|^\/$'
    ConstraintDescription: 'Role path must start and end with forward slash (/)'

  RoleManagedPolicy1:
    Type: String
```

```
    Description: Optional managed policy ARN 1
    Default: ""
  RoleManagedPolicy2:
    Type: String
    Description: Optional managed policy ARN 2
    Default: ""
  RoleManagedPolicy3:
    Type: String
    Description: Optional managed policy ARN 3
    Default: ""
  RoleManagedPolicy4:
    Type: String
    Description: Optional managed policy ARN 4
    Default: ""
  RoleManagedPolicy5:
    Type: String
    Description: Optional managed policy ARN 5
    Default: ""

Conditions:
  HasCustomRoleName: !Not [!Equals [!Ref RoleName, ""]]
  HasPermissionsBoundary: !Not [!Equals [!Ref RolePermissionsBoundary, ""]]
  HasPolicy1: !Not [!Equals [!Ref RoleManagedPolicy1, ""]]
  HasPolicy2: !Not [!Equals [!Ref RoleManagedPolicy2, ""]]
  HasPolicy3: !Not [!Equals [!Ref RoleManagedPolicy3, ""]]
  HasPolicy4: !Not [!Equals [!Ref RoleManagedPolicy4, ""]]
  HasPolicy5: !Not [!Equals [!Ref RoleManagedPolicy5, ""]]

Resources:
  SAMLFederatedRole:
    Type: 'AWS::IAM::Role'
    Properties:
      RoleName: !If
        - HasCustomRoleName
        - !Ref RoleName
        - !Ref AWS::NoValue
      Description: 'IAM role with SAML provider trust'
      MaxSessionDuration: !Ref RoleSessionDuration
      PermissionsBoundary: !If
        - HasPermissionsBoundary
        - !Ref RolePermissionsBoundary
        - !Ref AWS::NoValue
      Path: !Ref RolePath
      AssumeRolePolicyDocument:
```

```
        Version: '2012-10-17       '
        Statement:
          - Effect: Allow
            Principal:
              Federated: !Ref SAMLProviderARN
            Action: 'sts:AssumeRoleWithSAML'
            Condition:
              StringEquals:
                'SAML:aud': 'https://signin.aws.amazon.com/saml'
      ManagedPolicyArns:
        !Split
          - ','
          - !Join
            - ','
            - - !If [HasPolicy1, !Ref RoleManagedPolicy1, !Ref 'AWS::NoValue']
              - !If [HasPolicy2, !Ref RoleManagedPolicy2, !Ref 'AWS::NoValue']
              - !If [HasPolicy3, !Ref RoleManagedPolicy3, !Ref 'AWS::NoValue']
              - !If [HasPolicy4, !Ref RoleManagedPolicy4, !Ref 'AWS::NoValue']
              - !If [HasPolicy5, !Ref RoleManagedPolicy5, !Ref 'AWS::NoValue']

Outputs:
  RoleARN:
    Description: 'ARN of the created IAM Role'
    Value: !GetAtt SAMLFederatedRole.Arn
    Export:
      Name: !Sub '${AWS::StackName}-RoleARN'
```

IAM 教程：使用 Amazon CloudFormation 模板创建 SAML 身份提供者（IdP）和 SAML 联合身份验证 IAM 角色

要熟悉 SAML 联合身份验证及其功能，则需使用 Amazon CloudFormation 模板设置 SAML 身份提供者（IdP）和关联的联合身份验证 IAM 角色。本教程介绍如何在单个堆栈中同时创建两种资源。

该模板创建一个可用于联合访问 Amazon 资源的 SAML IdP，以及一个信任 SAML 提供者的 IAM 角色。经外部 IdP 进行身份验证的用户可代入此角色访问 Amazon 资源。

所部署的资源包括以下部分：

- 使用 IdP 的元数据文档配置的 SAML IdP。
- 信任 SAML IdP 的联合 IAM 角色，可由经过身份验证的用户代入此角色。

- 可配置的托管策略，可以附加到该角色以授予特定权限。

先决条件

本教程假定您已准备好以下各项：

- 本地计算机上安装了 Python 3.6 或更高版本，可运行本教程中用于对 IdP 的 SAML 元数据 XML 文件进行格式化的 Python 命令。
- 来自外部 IdP 的 SAML 元数据文档，另存为 XML 文件。

使用 Amazon CloudFormation 创建 SAML IdP 和角色

要创建 SAML IdP 和联合身份验证角色，您需要创建 CloudFormation 模板，并使用此模板创建包含这两种资源的堆栈。

创建模板

首先，创建 CloudFormation 模板。

1. 在 模板 部分中，单击 JSON 或 YAML 选项卡上的复制图标，以复制模板内容。
2. 将模板内容粘贴到新文件中。
3. 将该文件保存在本地。

创建堆栈

接下来，使用已保存的模板来预置 CloudFormation 堆栈。

1. 通过以下网址打开 Amazon CloudFormation 控制台：https://console.aws.amazon.com/cloudformation。
2. 在堆栈页面，从创建堆栈菜单中选择（采用新资源（标准））。
3. 指定模板：
    a. 在先决条件下，选择选择现有模板。
    b. 在指定模板下，选择上传模板文件。
    c. 选择选择文件，导航到模板文件并将其选中。
    d. 选择下一步。

4. 指定以下堆栈详细信息：

   a. 输入堆栈名称。

   b. 对于 IdentityProviderName，可将其留空以根据堆栈名称自动生成名称，也可以为 SAML IdP 输入自定义名称。

      示例：`CompanyIdP` 或 `EnterpriseSSO`

   c. 对于 IdentityProvidersamlMetadataDocument，在将 SAML 元数据 XML 文件粘贴到此字段之前，需要将其格式化为单行。这是必要的，因为 CloudFormation 控制台要求 XML 内容在通过控制台参数传递时需格式化为单行。

      使用以下 Python 命令重新格式化 XML 文件：

```
python3 -c "import sys, re; content=open(sys.argv[1]).read(); print(re.sub(r'>
\s+<', '><', content.replace('\n', '').replace('\r', '').strip()))" saml-
metadata.xml
```

> **Note**
>
> IdP 的 SAML 元数据文档必须格式化为单行才能输入控制台参数。Python 命令删除了换行符和多余空格以创建所需的格式，同时保留所有原始内容和结构。

      复制 Python 命令的输出并将其粘贴到 IdentityProviderSAMLMetadataDocument 字段中。

      格式化的 SAML 元数据文档示例（缩减版）：

```
<?xml version="1.0" encoding="UTF-8"?><md:EntityDescriptor
 xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
 entityID="https://portal.sso.example.com/saml/assertion/
CompanyIdP"><md:IDPSSODescriptor WantAuthnRequestsSigned="false"
 protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol"><md:KeyDescriptor
 use="signing"><ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/
xmldsig#"><ds:X509Data><ds:X509Certificate>MIIDXTCCAkWgAwIBAgIJAJC1HiIAZAiIMA0GCSqGSIb3
ds:X509Certificate></ds:X509Data></ds:KeyInfo></
md:KeyDescriptor><md:SingleLogoutService
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
 Location="https://portal.sso.example.com/saml/logout/
CompanyIdP"/><md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</md:NameIDFormat><md:SingleSignOnService
```

```
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" Location="https://
portal.sso.example.com/saml/assertion/CompanyIdP"/></md:IDPSSODescriptor></
md:EntityDescriptor>
```

d. 对于 RoleName，可将其留空以根据堆栈名称自动生成名称，也可以为联合 IAM 角色输入自定义名称。

   示例：`SAML-Developer-Access` 或 `SAML-ReadOnly-Role`

e. 对于其他参数，请接受默认值或根据需要输入自己的值：

   - IdentityProviderAddPrivateKey - 解密 SAML 断言的可选私钥
   - IdentityProviderAssertionEncryptionMode - SAML 断言加密模式

     示例值：`Allowed`，`Required`，或者留空表示不加密
   - RoleSessionDuration - 最大会话持续时间，以秒为单位（3600-43200，默认为 7200）

     示例：`14400`（4 小时）
   - RolePermissionsBoundary - 权限边界策略的可选 ARN

     示例：`arn:aws:iam::123456789012:policy/DeveloperBoundary`
   - RolePath - IAM 角色的路径（默认为 /）

     示例：`/saml-roles/`
   - RoleManagedPolicy1-5 - 最多 5 个待附加托管策略的可选 ARN

     RoleManagedPolicy1 示例：`arn:aws:iam::aws:policy/ReadOnlyAccess`

     RoleManagedPolicy2 示例：`arn:aws:iam::123456789012:policy/CustomPolicy`

f. 选择下一步。

5. 配置堆栈选项：

   a. 在堆栈故障选项下，选择删除所有新创建的资源。

   > **Note**
   >
   > 如果选择此选项，即使堆栈创建失败，您可能也不会因为删除策略规定保留资源而向资源付费。

   b. 接受其他所有默认值。

c. 在功能下勾选复选框，确认 CloudFormation 可能在您的账户中创建 IAM 资源。

d. 选择下一步。

6. 查看堆栈详细信息并选择提交。

Amazon CloudFormation 将创建堆栈。一旦堆栈创建完成之后，堆栈资源就即可投入使用。可使用堆栈详细信息页面上的资源选项卡来查看账户中预置的资源。

堆栈将输出以下值，可参见输出选项卡：

- RoleARN：已创建的 IAM 角色的 ARN（例如，`arn:aws:iam::123456789012:role/SAML-Developer-Access` 或 `arn:aws:iam::123456789012:role/stack-name-a1b2c3d4`，如果使用自动生成的名称）。
- IdentityProviderARN：已创建的 SAML IdP 的 ARN（例如，`arn:aws:iam::123456789012:saml-provider/CompanyIdP`）。

在配置 IdP 以发送相应 SAML 属性进行角色代入时，需要这两个 ARN。

测试 SAML 联合身份验证

一旦创建 SAML IdP 和联合角色后，可测试联合身份验证设置。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。

   您应该能在列表中看到新创建的 SAML IdP。
3. 选择 IdP 名称以查看其详细信息。

   在 IdP 详细信息页面上，可以看到 SAML 元数据文档和其他配置详细信息。
4. 在导航窗格中，选择角色。
5. 找到并选择新创建的联合角色。

   在角色详细信息页面上，可以看到允许 SAML IdP 代入此角色的信任策略。
6. 选择信任关系选项卡，查看信任策略。

   信任策略应表明在 SAML 受众 (`SAML:aud`) 匹配 `https://signin.aws.amazon.com/saml` 的条件下，信任该 SAML IdP 代入此角色。

清理：删除资源

最后一步，您将删除该堆栈及其包含的资源。

1. 打开 Amazon CloudFormation 控制台。
2. 在堆栈页面上，选择根据该模板创建的堆栈，选择删除，然后确认删除。

   CloudFormation 启动删除堆栈及其包含的所有资源。

CloudFormation 模板详细信息

资源

本教程的 Amazon CloudFormation 模板将在您的账户中创建以下资源：

- `AWS::IAM::SAMLProvider`：一个在 Amazon 和外部 IdP 之间建立信任的 SAML IdP。
- `AWS::IAM::Role`：一个可以由通过 SAML IdP 进行身份验证的用户代入的联合 IAM 角色。

配置

模板包含如下可配置参数：

- IdentityProviderName - SAML IdP 的名称（自动生成的名称请留空）
- IdentityProviderSAMLMetadataDocument - 来自 IdP 的 SAML 元数据文档（必需）
- IdentityProviderAddPrivateKey - 解密 SAML 断言的可选私钥
- IdentityProviderAssertionEncryptionMode - SAML 断言加密模式
- RoleName - IAM 角色的名称（自动生成的名称请留空）
- RolePath - IAM 角色的路径（默认为 /）
- RolePermissionsBoundary - 权限边界策略的可选 ARN
- RoleSessionDuration - 最大会话持续时间，以秒为单位（3600-43200，默认为 7200）
- RoleManagedPolicy1-5 - 最多 5 个待附加托管策略的可选 ARN

CloudFormation 模板

将以下 JSON 或 YAML 代码另存为单独文件，以用作本教程的 CloudFormation 模板。

JSON

```
{
  "AWSTemplateFormatVersion": "2010-09-09",
  "Description": "[AWSDocs] IAM: tutorial_saml-idp-and-federated-role",
  "Parameters": {
    "IdentityProviderName": {
      "Type": "String",
      "Description": "Name of the SAML Identity Provider (leave empty for auto-
generated name like '{StackName}-{UniqueId}')",
      "Default": "",
      "AllowedPattern": "^$|^[a-zA-Z0-9._-]+$",
      "ConstraintDescription": "Must be empty or contain only alphanumeric
 characters, periods, underscores, and hyphens"
    },
    "IdentityProviderSAMLMetadataDocument": {
      "Type": "String",
      "Description": "SAML metadata document from identity provider"
    },
    "IdentityProviderAddPrivateKey": {
      "Type": "String",
      "Description": "Optional private key for decrypting SAML assertions. The
 private key must be a .pem file that uses AES-GCM or AES-CBC encryption algorithm
 to decrypt SAML assertions.",
      "Default": ""
    },
    "IdentityProviderAssertionEncryptionMode": {
      "Type": "String",
      "Description": "Optional, sets encryption mode for SAML assertions",
      "Default": "",
      "AllowedValues": ["", "Allowed", "Required"]
    },
    "RoleName": {
      "Type": "String",
      "Description": "Name of the IAM Role (leave empty for auto-generated name like
 '{StackName}-{UniqueId}')",
      "Default": "",
      "AllowedPattern": "^$|^[\\w+=,.@-]{1,64}$",
      "ConstraintDescription": "Must be empty or 1-64 characters and can contain
 alphanumeric characters and +=,.@-"
    },
    "RolePath": {
      "Type": "String",
      "Description": "Path for the IAM Role",
```

```
      "AllowedPattern": "(^\\/$)|(^\\/.*\\/$)",
      "Default": "/"
    },
    "RolePermissionsBoundary": {
      "Type": "String",
      "Description": "Optional ARN of the permissions boundary policy (leave empty
for none)",
      "Default": ""
    },
    "RoleSessionDuration": {
      "Description": "The maximum session duration (in seconds) that you want to set
for the specified role (3600-43200)",
      "Type": "Number",
      "MinValue": 3600,
      "MaxValue": 43200,
      "Default": 7200
    },
    "RoleManagedPolicy1": {
      "Type": "String",
      "Description": "Optional managed policy ARN 1",
      "Default": ""
    },
    "RoleManagedPolicy2": {
      "Type": "String",
      "Description": "Optional managed policy ARN 2",
      "Default": ""
    },
    "RoleManagedPolicy3": {
      "Type": "String",
      "Description": "Optional managed policy ARN 3",
      "Default": ""
    },
    "RoleManagedPolicy4": {
      "Type": "String",
      "Description": "Optional managed policy ARN 4",
      "Default": ""
    },
    "RoleManagedPolicy5": {
      "Type": "String",
      "Description": "Optional managed policy ARN 5",
      "Default": ""
    }
  },
  "Conditions": {
```

```
   "HasCustomProviderName": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"IdentityProviderName"}, ""]}]},
   "HasCustomRoleName": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleName"}, ""]}]},
   "HasPermissionsBoundary": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"RolePermissionsBoundary"}, ""]}]},
   "HasPolicy1": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy1"},
""]}]},
   "HasPolicy2": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy2"},
""]}]},
   "HasPolicy3": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy3"},
""]}]},
   "HasPolicy4": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy4"},
""]}]},
   "HasPolicy5": {"Fn::Not": [{"Fn::Equals": [{"Ref": "RoleManagedPolicy5"},
""]}]},
   "HasPrivateKey": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"IdentityProviderAddPrivateKey"}, ""]}]},
   "HasAssertionEncryptionMode": {"Fn::Not": [{"Fn::Equals": [{"Ref":
"IdentityProviderAssertionEncryptionMode"}, ""]}]}
 },
 "Resources": {
   "SAMLProvider": {
     "Type": "AWS::IAM::SAMLProvider",
     "Properties": {
       "Name": {"Fn::If": ["HasCustomProviderName", {"Ref":
"IdentityProviderName"}, {"Ref": "AWS::NoValue"}]},
       "SamlMetadataDocument": {"Ref": "IdentityProviderSAMLMetadataDocument"},
       "AddPrivateKey": {"Fn::If": ["HasPrivateKey", {"Ref":
"IdentityProviderAddPrivateKey"}, {"Ref": "AWS::NoValue"}]},
       "AssertionEncryptionMode": {"Fn::If": ["HasAssertionEncryptionMode", {"Ref":
"IdentityProviderAssertionEncryptionMode"}, {"Ref": "AWS::NoValue"}]}
     }
   },
   "SAMLFederatedRole": {
     "Type": "AWS::IAM::Role",
     "Properties": {
       "RoleName": {"Fn::If": ["HasCustomRoleName", {"Ref": "RoleName"}, {"Ref":
"AWS::NoValue"}]},
       "Path": {"Ref": "RolePath"},
       "Description": "SAML federated IAM role for SSO access with specified
permissions",
       "MaxSessionDuration": {"Ref": "RoleSessionDuration"},
       "PermissionsBoundary": {"Fn::If": ["HasPermissionsBoundary", {"Ref":
"RolePermissionsBoundary"}, {"Ref": "AWS::NoValue"}]},
```

```
        "AssumeRolePolicyDocument": {
          "Version": "2012-10-17",
          "Statement": [
            {
              "Effect": "Allow",
              "Principal": {
                "Federated": {"Ref": "SAMLProvider"}
              },
              "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity",
                "sts:TagSession"
              ],
              "Condition": {
                "StringEquals": {
                  "SAML:aud": "https://signin.aws.amazon.com/saml"
                }
              }
            }
          ]
        },
        "ManagedPolicyArns": {
          "Fn::Split": [
            ",",
            {
              "Fn::Join": [
                ",",
                [
                  {"Fn::If": ["HasPolicy1", {"Ref": "RoleManagedPolicy1"}, {"Ref":
"AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy2", {"Ref": "RoleManagedPolicy2"}, {"Ref":
"AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy3", {"Ref": "RoleManagedPolicy3"}, {"Ref":
"AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy4", {"Ref": "RoleManagedPolicy4"}, {"Ref":
"AWS::NoValue"}]},
                  {"Fn::If": ["HasPolicy5", {"Ref": "RoleManagedPolicy5"}, {"Ref":
"AWS::NoValue"}]}
                ]
              ]
            }
          ]
        }
      }
```

```
    }
  },
  "Outputs": {
    "RoleARN": {
      "Description": "ARN of the created IAM Role",
      "Value": {"Fn::GetAtt": ["SAMLFederatedRole", "Arn"]},
      "Export": {
        "Name": {"Fn::Sub": "${AWS::StackName}-RoleARN"}
      }
    },
    "IdentityProviderARN": {
      "Description": "ARN of the created SAML Identity Provider",
      "Value": {"Ref": "SAMLProvider"},
      "Export": {
        "Name": {"Fn::Sub": "${AWS::StackName}-IdentityProviderARN"}
      }
    }
  }
}
```

YAML

```
AWSTemplateFormatVersion: '2010-09-09'
Description: '[AWSDocs] IAM: tutorial_saml-idp-and-federated-role'

Parameters:
  IdentityProviderName:
    Type: String
    Description: Name of the SAML Identity Provider (leave empty for auto-generated
 name like '{StackName}-{UniqueId}')
    Default: ""
    AllowedPattern: '^$|^[a-zA-Z0-9._-]+$'
    ConstraintDescription: Must be empty or contain only alphanumeric characters,
 periods, underscores, and hyphens

  IdentityProviderSAMLMetadataDocument:
    Type: String
    Description: SAML metadata document from identity provider

  IdentityProviderAddPrivateKey:
    Type: String
```

```
    Description: Optional private key for decrypting SAML assertions. The private
key must be a .pem file that uses AES-GCM or AES-CBC encryption algorithm to
decrypt SAML assertions.
    Default: ""

  IdentityProviderAssertionEncryptionMode:
    Type: String
    Description: Optional, sets encryption mode for SAML assertions
    Default: ""
    AllowedValues:
      - ""
      - "Allowed"
      - "Required"

  RoleName:
    Type: String
    Description: Name of the IAM Role (leave empty for auto-generated name like
'{StackName}-{UniqueId}')
    Default: ""
    AllowedPattern: '^$|^[\w+=,.@-]{1,64}$'
    ConstraintDescription: "Must be empty or 1-64 characters and can contain
alphanumeric characters and +=,.@-"

  RolePath:
    Type: String
    Description: Path for the IAM Role
    AllowedPattern: (^\/$)|(^\/.*\/$)
    Default: "/"

  RolePermissionsBoundary:
    Type: String
    Description: Optional ARN of the permissions boundary policy (leave empty for
none)
    Default: ""

  RoleSessionDuration:
    Description: The maximum session duration (in seconds) that you want to set for
the specified role (3600-43200)
    Type: Number
    MinValue: 3600
    MaxValue: 43200
    Default: 7200

  RoleManagedPolicy1:
```

```
    Type: String
    Description: Optional managed policy ARN 1
    Default: ""
  RoleManagedPolicy2:
    Type: String
    Description: Optional managed policy ARN 2
    Default: ""
  RoleManagedPolicy3:
    Type: String
    Description: Optional managed policy ARN 3
    Default: ""
  RoleManagedPolicy4:
    Type: String
    Description: Optional managed policy ARN 4
    Default: ""
  RoleManagedPolicy5:
    Type: String
    Description: Optional managed policy ARN 5
    Default: ""

Conditions:
  HasCustomProviderName: !Not [!Equals [!Ref IdentityProviderName, ""]]
  HasCustomRoleName: !Not [!Equals [!Ref RoleName, ""]]
  HasPermissionsBoundary: !Not [!Equals [!Ref RolePermissionsBoundary, ""]]
  HasPolicy1: !Not [!Equals [!Ref RoleManagedPolicy1, ""]]
  HasPolicy2: !Not [!Equals [!Ref RoleManagedPolicy2, ""]]
  HasPolicy3: !Not [!Equals [!Ref RoleManagedPolicy3, ""]]
  HasPolicy4: !Not [!Equals [!Ref RoleManagedPolicy4, ""]]
  HasPolicy5: !Not [!Equals [!Ref RoleManagedPolicy5, ""]]
  HasPrivateKey: !Not [!Equals [!Ref IdentityProviderAddPrivateKey, ""]]
  HasAssertionEncryptionMode: !Not [!Equals [!Ref
 IdentityProviderAssertionEncryptionMode, ""]]

Resources:
  SAMLProvider:
    Type: AWS::IAM::SAMLProvider
    Properties:
      Name: !If
        - HasCustomProviderName
        - !Ref IdentityProviderName
        - !Ref AWS::NoValue
      SamlMetadataDocument: !Ref IdentityProviderSAMLMetadataDocument
      AddPrivateKey: !If
        - HasPrivateKey
```

```
        - !Ref IdentityProviderAddPrivateKey
        - !Ref AWS::NoValue
      AssertionEncryptionMode: !If
        - HasAssertionEncryptionMode
        - !Ref IdentityProviderAssertionEncryptionMode
        - !Ref AWS::NoValue

  SAMLFederatedRole:
    Type: AWS::IAM::Role
    Properties:
      RoleName: !If
        - HasCustomRoleName
        - !Ref RoleName
        - !Ref AWS::NoValue
      Path: !Ref RolePath
      Description: "SAML federated IAM role for SSO access with specified
permissions"
      MaxSessionDuration: !Ref RoleSessionDuration
      PermissionsBoundary: !If
        - HasPermissionsBoundary
        - !Ref RolePermissionsBoundary
        - !Ref AWS::NoValue
      AssumeRolePolicyDocument:
        Version: '2012-10-17       '
        Statement:
          - Effect: Allow
            Principal:
              Federated: !Ref SAMLProvider
            Action:
              - 'sts:AssumeRole'
              - 'sts:SetSourceIdentity'
              - 'sts:TagSession'
            Condition:
              StringEquals:
                'SAML:aud': 'https://signin.aws.amazon.com/saml'
      ManagedPolicyArns: !Split
        - ','
        - !Join
          - ','
          - - !If [HasPolicy1, !Ref RoleManagedPolicy1, !Ref "AWS::NoValue"]
            - !If [HasPolicy2, !Ref RoleManagedPolicy2, !Ref "AWS::NoValue"]
            - !If [HasPolicy3, !Ref RoleManagedPolicy3, !Ref "AWS::NoValue"]
            - !If [HasPolicy4, !Ref RoleManagedPolicy4, !Ref "AWS::NoValue"]
            - !If [HasPolicy5, !Ref RoleManagedPolicy5, !Ref "AWS::NoValue"]
```

```
Outputs:
  RoleARN:
    Description: ARN of the created IAM Role
    Value: !GetAtt SAMLFederatedRole.Arn
    Export:
      Name: !Sub '${AWS::StackName}-RoleARN'

  IdentityProviderARN:
    Description: ARN of the created SAML Identity Provider
    Value: !Ref SAMLProvider
    Export:
      Name: !Sub '${AWS::StackName}-IdentityProviderARN'
```

IAM 身份

IAM 身份可以与一个或多个策略相关联，其确定身份可在何种条件下对哪些 Amazon 资源执行哪些操作。IAM 身份包括 IAM 用户、IAM 组和 IAM 角色。IAM 身份是一种身份类型，代表人类用户或编程工作负载，可对身份进行验证，然后向其授予在 Amazon Web Services 账户中执行操作的权限。IAM 身份包括 IAM 用户和 IAM 角色。有关常用术语的定义，请参阅术语。

您可以通过外部身份提供商对现有身份进行联合身份验证。此类身份将代入 IAM 角色来访问 Amazon 资源。有关更多信息，请参阅 the section called “身份提供程序和 Amazon 中的联合身份验证”。

您还可以使用 Amazon IAM Identity Center 创建和管理身份以及访问 Amazon 资源。IAM Identity Center 权限集会自动创建提供资源访问权限所需的 IAM 角色。有关更多信息，请参阅 What is IAM Identity Center?

Amazon Web Services 账户根用户 是在您在建立 Amazon Web Services 账户 时创建的 Amazon Web Services 账户 主体。根用户有权访问该账户中的所有 Amazon 服务和资源。有关更多信息，请参阅 the section called “IAM 根用户”。

> **Note**
>
> - 使用 IAM 身份时，请遵循 IAM 中的安全最佳实践。
> - 使用根用户时，请遵循适合 Amazon Web Services 账户 的根用户最佳实践。
> - 如果您在登录时遇到问题，请参阅 Sign in to the Amazon Web Services 管理控制台。

IAM 根用户

当您首次创建 Amazon Web Services 账户时，最初使用的是一个对账户中所有 Amazon Web Services 服务和资源拥有完全访问权限的登录身份。此身份称作 Amazon Web Services 账户根用户。有关更多信息，请参阅 Amazon 账户根用户。

IAM 用户

IAM 用户是 Amazon Web Services 账户内对某个人员或应用程序具有特定权限的一个身份。有关更多信息，请参阅 IAM 用户。

IAM 用户组

IAM 用户组是一个指定一组 IAM 用户的身份。有关更多信息，请参阅用户组。

IAM 角色

IAM 角色是 Amazon Web Services 账户中具有特定权限的身份。它类似于 IAM 用户，但与特定人员没有关联。有关更多信息，请参阅 IAM 角色。

Amazon Web Services 账户根用户

当您首次创建 Amazon Web Services (Amazon) 账户时，最初使用的是一个对账户中所有 Amazon 服务和资源具有完全访问权限的单点登录身份。此身份称作 Amazon 账户根用户。您用于创建 Amazon Web Services 账户 的电子邮件地址和密码是您以根用户登录时使用的凭证。

- 仅使用根用户执行需要根级别权限的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅需要根用户凭证的任务。
- 请遵循您的 Amazon Web Services 账户 的根用户最佳实践。
- 如果您在登录时遇到问题，请参阅 Sign in to the Amazon Web Services 管理控制台。

> **Important**
>
> 在北京和宁夏区域，没有根用户的概念。所有用户都是 IAM 用户，包括创建 Amazon 账户的用户。

> **Important**
>
> 强烈建议您不要使用根用户来执行日常任务，并遵循您的 Amazon Web Services 账户 的根用户的最佳实践。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅 需要根用户凭证的任务。

虽然默认情况下将对根用户强制执行 MFA，但需要客户在初始账户创建期间，或根据登录时的提示进行操作才能添加 MFA。有关使用 MFA 保护根用户的更多信息，请参阅 Amazon Web Services 账户根用户 的多重身份验证。

集中管理成员账户的根访问权限

为了帮助您大规模管理凭证，您可以在 Amazon Organizations 中集中保护对成员账户的根用户凭证的访问。启用 Amazon Organizations 后，您可以将所有 Amazon 账户合并到一个组织中进行集中管理。通过集中根访问，您可以移除根用户凭证，并对成员账户执行以下特权任务。

移除成员账户的根用户证书

集中成员账户的根访问后，您可以选择从 Organizations 的成员账户中删除根用户凭证。您可以移除根用户密码、访问密钥、签名证书，并停用多重身份验证（MFA）。默认情况下，您在 Organizations 中创建的新账户不具有根用户证书。除非启用账户恢复，否则成员账户不能登录到他们的根用户或为其根用户执行密码恢复。

执行需要根用户凭证的特权任务

有些任务只能在您以账户的根用户身份登录时执行。其中一些 需要根用户凭证的任务 可以由管理账户或 IAM 的委派管理员执行。要了解有关对成员账户采取特权操作的更多信息，请参阅执行特权任务。

启用根用户的账户恢复

如果您需要恢复成员账户的根用户凭证，则 Organizations 管理账户或委派管理员可以执行允许密码恢复特权任务。有权访问成员账户的根用户电子邮件收件箱的人可以重置根用户密码，以恢复根用户证书。建议您在完成需要访问根用户的任务后删除根用户凭证。

集中成员账户的根访问

根用户凭证是分配给具有对账户中所有 Amazon 服务和资源的完全访问权限的每个 Amazon Web Services 账户的初始凭证。启用 Amazon Organizations 后，您可以将所有 Amazon 账户合并到一个组织中进行集中管理。每个成员账户都有自己的根用户，该用户拥有在成员账户中执行任何操作的默认权限。建议您集中保护使用 Amazon Organizations 管理的 Amazon Web Services 账户的根用户凭证，以防止大规模的根用户凭证恢复和访问。

集中根访问后，您可以选择从组织的成员账户中删除根用户凭证。您可以移除根用户密码、访问密钥、签名证书，并停用多重身份验证（MFA）。默认情况下，您在 Amazon Organizations 中创建的新账户不具有根用户凭证。成员账户不能登录到他们的根用户或为其根用户执行密码恢复。

> **Note**
>
> 有些需要根用户凭证的任务可以由 IAM 的管理账户或委托管理员来执行，而有些任务只能在您以账户根用户登录时执行。

如果需要恢复成员账户的根用户凭证来执行其中一项任务，请遵照 执行特权任务 中的步骤，然后选择允许找回密码。然后有权访问该成员账户的根用户电子邮件收件箱的人可以按照步骤重置根用户密码，并登录到成员账户根用户。
建议您在完成需要访问根用户的任务后删除根用户凭证。

先决条件

在集中根访问之前，必须为账户配置以下设置：

- 您必须具有以下 IAM 权限：
  - `iam:GetAccessKeyLastUsed`
  - `iam:GetAccountSummary`
  - `iam:GetLoginProfile`
  - `iam:GetUser`
  - `iam:ListAccessKeys`
  - `iam:ListMFADevices`
  - `iam:ListSigningCertificates`
  - `sts:AssumeRoot`

  Note

  要审计成员账户的根用户凭证状态时，可在对 Amazon Organizations 成员账户执行特权任务时使用 IAMAuditRootUserCredentials Amazon 托管策略缩小权限范围，或使用任何具有 `iam:GetAccountSummary` 访问权限的策略。
  要生成根用户凭证信息报告，其他策略只需要执行 `iam:GetAccountSummary` 操作即可生成相同的输出。您还可以列出或获取个人根用户凭证信息，包括：
  - 是否有根用户密码
  - 是否有根用户访问密钥以及上次使用该密钥的时间
  - 如果根用户关联了签名证书
  - 根用户关联的 MFA 设备
  - 统一的根用户凭证状态列表
- 必须在 Amazon Organizations 中管理您的 Amazon Web Services 账户。
- 您必须具有以下权限才能在组织中启用此功能：

  - `iam:EnableOrganizationsRootCredentialsManagement`
  - `iam:EnableOrganizationsRootSessions`
  - `iam:ListOrganizationsFeatures`
  - `organizations:EnableAwsServiceAccess`
  - `organizations:ListAccountsForParent`
  - `organizations:RegisterDelegatedAdministrator`
- 为确保获得最佳控制台功能，建议启用以下附加权限：
  - `organizations:DescribeAccount`
  - `organizations:DescribeOrganization`
  - `organizations:ListAWSServiceAccessForOrganization`
  - `organizations:ListDelegatedAdministrators`
  - `organizations:ListOrganizationalUnitsForParent`
  - `organizations:ListParents`
  - `organizations:ListTagsForResource`

启用集中的根访问（控制台）

要在 Amazon Web Services 管理控制台中为成员账户启用此功能

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择根访问权限管理，然后选择启用。

> **Note**
>
> 如果您看到已禁用根访问权限管理，请在 Amazon Identity and Access Management 中启用 Amazon Organizations 的可信访问。有关详细信息，请参阅《Amazon Organizations 用户指南》中的 Amazon IAM 和 Amazon Organizations。

3. 在“要启用的功能”部分，选择要启用的功能。
   - 选择根凭证管理，以允许管理账户和 IAM 的委派管理员删除成员账户的根用户凭证。您必须在成员账户中启用特权根操作，以允许成员账户在删除其根用户凭证后恢复这些凭证。

- 选择成员账户中的特权根操作，以允许管理账户和 IAM 的委派管理员执行某些需要根用户凭证的任务。

4. （可选）输入获得授权管理根用户访问并对成员账户采取特权措施的委派管理员的账户 ID。建议使用用于安全或管理目的的账户。
5. 请选择启用。

启用集中的根访问（Amazon CLI）

要从 Amazon Command Line Interface（Amazon CLI）启用集中式根用户访问权限

1. 如果您尚未在 Amazon Organizations 中启用 Amazon Identity and Access Management 的可信访问权限，则请使用以下命令：aws organizations enable-aws-service-access。
2. 使用以下命令允许管理账户和委派管理员删除成员账户的根用户凭证：aws iam enable-organizations-root-credentials-management。
3. 使用以下命令允许管理账户和委派管理员执行需要根用户凭证的特定任务：aws iam enable-organizations-root-sessions。
4. （可选）使用以下命令注册委派管理员：aws organizations register-delegated-administrator。

   以下示例将账户 111111111111 分配为 IAM 服务的委派管理员。

```
aws organizations register-delegated-administrator
--service-principal iam.amazonaws.com
--account-id 111111111111
```

启用集中的根访问（Amazon API）

要从 Amazon API 启用集中式根用户访问权限

1. 如果您尚未在 Amazon Organizations 中启用 Amazon Identity and Access Management 的可信访问权限，则请使用以下命令：EnableAWSServiceAccess。
2. 使用以下命令允许管理账户和委派管理员删除成员账户的根用户凭证：EnableOrganizationsRootCredentialsManagement。
3. 使用以下命令允许管理账户和委派管理员执行需要根用户凭证的特定任务：EnableOrganizationsRootSessions。
4. （可选）使用以下命令注册委派管理员：RegisterDelegatedAdministrator。

后续步骤

集中保护组织中成员账户的特权凭证后，请参阅执行特权任务以对成员账户采取特权操作。

在 Amazon Organizations 成员账户上执行特权任务

Amazon Organizations 管理账户或IAM的委托管理员账户，可以在成员账户上执行一些原本需要根用户凭证的特权任务。通过集中式根访问权限，这些任务通过短期特权会话执行。这些会话提供限于特定特权操作的临时凭证，无需根用户登录成员账户。

启动特权会话后，您可以删除配置错误的 Amazon S3 存储桶策略、删除配置错误的 Amazon SQS 队列策略、删除成员账户的根用户凭证，以及重新启用成员账户的根用户凭证。

> **Note**
>
> 要使用集中式根访问权限，您必须通过管理账户或委托管理员账户，以被明确授予 `sts:AssumeRoot` 权限的 IAM 用户或角色身份登录。您可以使用根用户凭证来调用 `sts:AssumeRoot`。

先决条件

在启动特权会话之前，您必须具备以下设置：

- 您已在组织中启用集中根访问。有关启用此功能的步骤，请参阅集中成员账户的根访问。
- 您的管理账户或委派管理员账户具有以下权限：`sts:AssumeRoot`

对成员账户执行特权操作（控制台）

要在 Amazon Web Services 管理控制台中为成员账户的特权操作启动会话

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择根访问管理。
3. 从成员账户列表中选择一个名称，然后选择采取特权操作。
4. 选择您想要在成员账户中执行的特权操作。
   - 选择删除 Amazon S3 存储桶策略，以删除一项配置错误的存储桶策略，此策略将会拒绝所有主体访问 Amazon S3 存储桶策略。

a. 选择浏览 S3，从成员账户拥有的存储桶中选择一个名称，然后选择选择。

b. 选择删除存储桶策略。

c. 在删除配置错误的策略后，使用 Amazon S3 控制台来纠正存储桶策略。有关更多信息，请参阅《Amazon S3 用户指南》中的使用 Amazon S3 控制台添加存储桶策略。

- 选择删除 Amazon SQS 策略，以删除 Amazon Simple Queue Service 基于资源的策略，此策略将会拒绝所有主体访问 Amazon SQS 队列。

  a. 在 SQS 队列名称中输入队列名称，然后选择删除 SQS 策略。

  b. 在删除配置错误的策略后，使用 Amazon SQS 控制台来纠正队列策略。有关更多信息，请参阅《Amazon SQS 开发人员指南》中的 Configuring an access policy in Amazon SQS。

- 选择删除根凭证，以删除成员账户的根访问权限。删除根用户凭证将会移除根用户密码、访问密钥、签名证书并停用成员账户的多重身份验证（MFA）。

  - 选择删除根凭证。

- 选择允许密码恢复，以恢复成员账户的根用户凭证。

  仅当成员账户没有根用户凭证时，此选项才可用。

  a. 选择允许恢复密码。

  b. 执行此特权操作后，有权访问该成员账户的根用户电子邮件收件箱的人可以重置根用户密码并登录到成员账户根用户。

对成员账户执行特权操作（Amazon CLI）

要从 Amazon Command Line Interface 中为成员账户的特权操作启动会话

1. 使用以下命令假设根用户会话：aws sts assume-root。

> **Note**
>
> 全局端点不支持 `sts:AssumeRoot`。您必须将此请求发送到区域的 Amazon STS 端点。有关更多信息，请参阅 管理 Amazon Web Services 区域中的 Amazon STS。

在为成员账户启动特权根用户会话时，必须定义 `task-policy-arn`，以将会话的范围限定到会话期间要执行的特权操作。您可以使用以下 Amazon 托管策略之一来限定特权会话操作的范围。

- IAMAuditRootUserCredentials
- IAMCreateRootUserPassword
- IAMDeleteRootUserCredentials
- S3UnlockBucketPolicy
- SQSUnlockQueuePolicy

要限制管理账户或委派管理员在特权根用户会话期间可以执行的操作，您可以使用 Amazon STS 条件键 sts:TaskPolicyArn。

在以下示例中，委派管理员担任根用户来删除成员账户 ID *111122223333* 的根用户凭证。

```
aws sts assume-root \
  --target-principal 111122223333 \
  --task-policy-arn arn=arn:aws:iam::aws:policy/root-task/
IAMDeleteRootUserCredentials \
  --duration-seconds 900
```

2. 使用响应中的 `SessionToken`、`AccessKeyId` 和 `SecretAccessKey` 在成员账户中执行特权操作。您可以省略请求中的用户名和密码，以默认设置为成员账户。
   - 检查根用户凭证的状态。使用以下命令检查成员账户的根用户凭证状态。
     - get-user
     - get-login-profile
     - list-access-keys
     - list-signing-certificates
     - list-mfa-devices
     - get-access-key-last-used
   - 删除根用户凭证。使用以下命令删除根访问。您可以移除根用户密码、访问密钥、签名证书，并停用多重身份验证（MFA），以移除对根用户的所有访问及根用户的恢复。
     - delete-login-profile
     - delete-access-key

  - delete-signing-certificate
  - deactivate-mfa-device
- 删除 Amazon S3 存储桶策略。使用以下命令读取、编辑和删除配置错误的存储桶策略，该策略拒绝所有主体访问 Amazon S3 存储桶。
  - list-buckets
  - get-bucket-policy
  - put-bucket-policy
  - delete-bucket-policy
- 删除 Amazon SQS 策略。使用以下命令查看并删除将会拒绝所有主体访问 Amazon SQS 队列的 Amazon Simple Queue Service 基于资源的策略。
  - list-queues
  - get-queue-url
  - get-queue-attributes
  - set-queue-attributes
- 允许密码恢复。使用以下命令查看用户名并恢复成员账户的根用户凭证。
  - get-login-profile
  - create-login-profile

对成员账户执行特权操作（Amazon API）

要从 Amazon API 中为成员账户的特权操作启动会话

1. 使用以下命令假设根用户会话：AssumeRoot。

> **Note**
>
> 全局端点不支持 AssumeRoot。您必须将此请求发送到区域的 Amazon STS 端点。有关更多信息，请参阅 管理 Amazon Web Services 区域中的 Amazon STS。

在为成员账户启动特权根用户会话时，必须定义 `TaskPolicyArn`，以将会话的范围限定到会话期间要执行的特权操作。您可以使用以下 Amazon 托管策略之一来限定特权会话操作的范围。

- IAMAuditRootUserCredentials
- IAMCreateRootUserPassword
- IAMDeleteRootUserCredentials
- S3UnlockBucketPolicy
- SQSUnlockQueuePolicy

要限制管理账户或委派管理员在特权根用户会话期间可以执行的操作，您可以使用 Amazon STS 条件键 sts:TaskPolicyArn。

在以下示例中，委派管理员担任根用户来读取、编辑和删除成员账户 ID *111122223333* 的 Amazon S3 存储桶中配置错误的基于资源的策略。

```
https://sts.us-east-2.amazonaws.com/
  ?Version=2011-06-15
  &Action=AssumeRoot
  &TargetPrincipal=111122223333
  &PolicyArns.arn=arn:aws:iam::aws:policy/root-task/S3UnlockBucketPolicy
  &DurationSeconds=900
```

2. 使用响应中的 SessionToken、AccessKeyId 和 SecretAccessKey 在成员账户中执行特权操作。您可以省略请求中的用户名和密码，以默认设置为成员账户。
   - 检查根用户凭证的状态。使用以下命令检查成员账户的根用户凭证状态。
     - GetUser
     - GetLoginProfile
     - ListAccessKeys
     - ListSigningCertificates
     - ListMFADevices
     - GetAccessKeyLastUsed
   - 删除根用户凭证。使用以下命令删除根访问。您可以移除根用户密码、访问密钥、签名证书，并停用多重身份验证（MFA），以移除对根用户的所有访问及根用户的恢复。
     - DeleteLoginProfile
     - DeleteAccessKey
     - DeleteSigningCertificate

    - DeactivateMfaDevice
- 删除 Amazon S3 存储桶策略。使用以下命令读取、编辑和删除配置错误的存储桶策略，该策略拒绝所有主体访问 Amazon S3 存储桶。
    - ListBuckets
    - GetBucketPolicy
    - PutBucketPolicy
    - DeleteBucketPolicy
- 删除 Amazon SQS 策略。使用以下命令查看并删除将会拒绝所有主体访问 Amazon SQS 队列的 Amazon Simple Queue Service 基于资源的策略。
    - ListQueues
    - GetQueueUrl
    - GetQueueAttributes
    - SetQueueAttributes
- 允许密码恢复。使用以下命令查看用户名并恢复成员账户的根用户凭证。
    - GetLoginProfile
    - CreateLoginProfile

Amazon Web Services 账户根用户 的多重身份验证

> **Important**
>
> Amazon 建议您在登录 Amazon 时，尽可能使用通行密钥或安全密钥，因为它们更能抵御网络钓鱼等攻击。有关更多信息，请参阅 通行密钥和安全密钥。

多重身份验证 (MFA) 是一种用于增强安全性的简单而有效的机制。第一个因素（密码）是您记住的秘密，也称为知识因素。其他因素可以是拥有因素（您拥有的东西，例如安全密钥）或固有因素（您与生俱来的东西，例如生物识别扫描）。为增强安全性，我们强烈建议您配置多重身份验证（MFA）以帮助保护 Amazon 资源。

> **Note**
>
> 所有 Amazon Web Services 账户 账户类型（独立账户、管理账户和成员账户）都需要为其根用户配置 MFA。如果尚未启用 MFA，则用户必须在首次登录尝试访问 Amazon Web Services 管理控制台后 35 天内注册 MFA。

您可以为 Amazon Web Services 账户根用户 和 IAM 用户启用 MFA。当您为根用户启用 MFA 时，它仅影响根用户凭证。有关如何为 IAM 用户启用 MFA 的更多信息，请参阅 IAM 中的 Amazon 多重身份验证。

> **Note**
>
> 使用 Amazon Organizations 托管的 Amazon Web Services 账户可以选择集中管理成员账户的根访问权限，以防止凭证恢复和大规模访问。如果启用此选项，则可以从成员账户中删除根用户凭证，包括密码和 MFA，从而有效地阻止以根用户身份登录、进行密码恢复或设置 MFA。或者，如果您希望维护基于密码的登录方法，请通过注册 MFA 来保护您的账户，以增强账户保护。

为您的根用户启用 MFA 之前，请查看并更新您的账户设置和联系信息，以确保您可以访问电子邮件和电话号码。如果您的 MFA 设备丢失、被盗或无法工作，您仍可以通过使用相应的电子邮件和电话号码验证您的身份，以作为根用户登录。要了解如何使用替代的身份验证因素登录，请参阅 在 IAM 中恢复受 MFA 保护的身份。要禁用此功能，请联系 Amazon Web Services 支持。

Amazon 为根用户支持以下 MFA 类型：

- 通行密钥和安全密钥
- 虚拟身份验证器应用程序
- 硬件 TOTP 令牌

通行密钥和安全密钥

Amazon Identity and Access Management 对 MFA 支持通行密钥和安全密钥。基于 FIDO 标准，通行密钥使用公有密钥加密技术来提供比密码更安全的强大防网络钓鱼身份验证。Amazon 支持两种类型的通行密钥：设备绑定通行密钥（安全密钥）和同步通行密钥。

- 安全密钥：这些是物理设备，例如 YubiKey，用作身份验证的第二个因素。单个安全密钥可以支持多个根用户账户和 IAM 用户。
- 同步通行密钥：它们使用来自 Google、Apple、Microsoft 账户等提供商的凭证管理器以及第三方服务（例如 1Password、Dashlane 和 Bitwarden）作为第二个因素。

您可以使用内置生物识别身份验证器（例如，Apple MacBook 上的 Touch ID）来解锁凭证管理器并登录 Amazon。通行密钥是通过您选择的提供商使用您的指纹、面部或设备 PIN 创建的。您还可以使用一台设备（如移动设备或硬件安全密钥）上的跨设备身份验证 (CDA) 密钥在另一台设备（如笔记本电脑）上登录。有关更多信息，请参阅跨设备身份验证 (CDA)。

您可以跨设备同步通行密钥以方便登录 Amazon，从而增强可用性和可恢复性。有关启用通行密钥和安全密钥的更多信息，请参阅 为根用户启用通行密钥或安全密钥（控制台）。

FIDO 联盟维护一份与 FIDO 规范兼容的所有经 FIDO 认证产品的列表。

虚拟身份验证器应用程序

虚拟身份验证器应用程序在电话或其他设备上运行，并模拟物理设备。虚拟身份验证器应用程序采用基于时间的一次性密码（TOTP）算法，并支持单个设备上的多个令牌。在登录期间，用户必须在出现提示时从该设备键入有效代码。分配给用户的每个令牌必须是唯一的。用户无法从另一个用户的令牌键入代码来进行身份验证。

我们建议您在等待硬件购买批准或等待硬件到达时使用虚拟 MFA 设备。有关可用作虚拟 MFA 设备的一些受支持应用程序的列表，请参阅多重身份验证（MFA）。有关使用 Amazon 设置虚拟 MFA 设备的说明，请参阅 为根用户启用虚拟 MFA 设备（控制台）。

硬件 TOTP 令牌

硬件设备以基于时间的一次性密码（TOTP）算法为基础生成六位数字代码。在登录时，用户必须在另一个网页上键入来自该设备的有效代码。分配给用户的每台 MFA 设备必须是唯一的。用户无法从另一个用户的设备键入代码来进行身份验证。有关受支持硬件 MFA 设备的信息，请参阅多重身份验证（MFA）。有关使用 Amazon 设置硬件 TOTP 令牌的说明，请参阅 为根用户启用硬件 TOTP 令牌（控制台）。

如果想使用物理 MFA 设备，我们建议使用 FIDO 安全密钥来代替硬件 TOTP 设备。FIDO 安全密钥具有无需电池、可抵御网络钓鱼的优点，并且支持在单台设备上使用多个根用户和 IAM 用户，从而增强安全性。

主题

- 为根用户启用通行密钥或安全密钥（控制台）
- 为根用户启用虚拟 MFA 设备（控制台）
- 为根用户启用硬件 TOTP 令牌（控制台）

为根用户启用通行密钥或安全密钥（控制台）

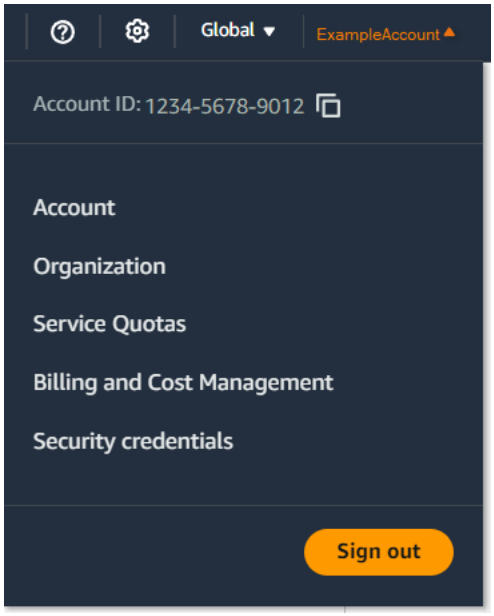
您只能从 Amazon Web Services 管理控制台 为根用户配置和启用通行密钥，而不能从 Amazon CLI 或 Amazon API 配置和启用。

为您的根用户启用通行密钥或安全密钥（控制台）

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 用户指南》中的 以根用户身份登录 Amazon Web Services 管理控制台。

2. 在导航栏的右侧，选择您的账户名称，然后选择 Security Credentials（安全凭证）。



3. 在您的根用户我的安全凭证页面的多重身份验证（MFA）下，选择分配 MFA 设备。
4. 在 MFA 设备名称页面上，输入设备名称，选择通行密钥或安全密钥，然后选择下一步。
5. 在设置设备上，设置您的通行密钥。使用面部或指纹等生物识别数据、设备 PIN 码或将 FIDO 安全密钥插入计算机的 USB 端口并点按即可创建通行密钥。
6. 按照浏览器上的说明选择通行密钥提供商或想要存储通行密钥的位置，以便在设备上使用。
7. 选择继续。

现在，您已经注册了用于 Amazon 的通行密钥。下次使用根用户凭证登录时，您必须使用您的通行密钥进行身份验证才能完成登录过程。

有关对 FIDO 安全密钥问题进行故障排除的帮助信息，请参阅 排查通行密钥和 FIDO 安全密钥问题。

为根用户启用虚拟 MFA 设备（控制台）

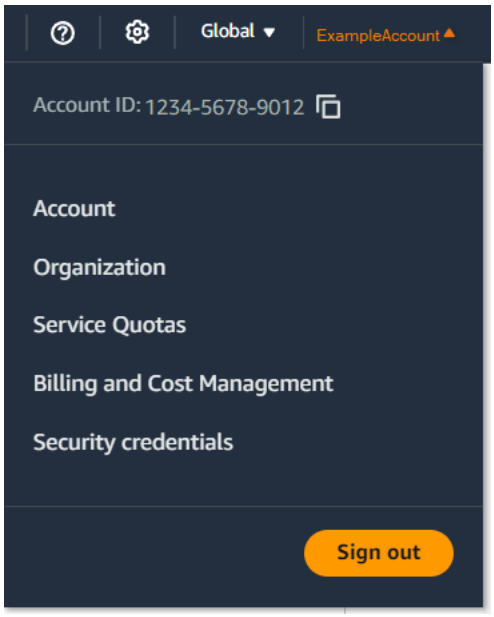
您可以使用 Amazon Web Services 管理控制台 为根用户配置和启用虚拟 MFA 设备。要为 Amazon Web Services 账户 启用 MFA 设备，您必须使用根用户凭证登录 Amazon。

配置和启用虚拟 MFA 设备以用于您的根用户（控制台）

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 用户指南》中的 以根用户身份登录 Amazon Web Services 管理控制台。

2. 在导航栏右侧，选择您的账户名称，然后选择 Security Credentials（安全凭证）。



3. 在多重身份验证 (MFA) 部分中，选择分配 MFA 设备。
4. 在向导中，键入设备名称，选择身份验证器应用程序，然后选择下一步。

   IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。

5. 在设备上打开虚拟 MFA 应用程序。

如果虚拟 MFA 应用程序支持多个虚拟 MFA 设备或账户，请选择相应的选项以创建新的虚拟 MFA 设备或账户。

6. 要配置应用程序，最简单的方法是使用应用程序扫描 QR 代码。如果您无法扫描代码，则可手动键入配置信息。IAM 生成的二维码和私有配置密钥与您的 Amazon Web Services 账户 关联，不能用于其他账户。但是，如果您失去对原始 MFA 设备的访问权，可以重新使用它们为您的账户配置新的 MFA 设备。

   - 要使用 QR 代码配置虚拟 MFA 设备，请在向导中，选择 Show QR code (显示 QR 代码)。然后，按照应用程序说明扫描代码。例如，您可能需要选择摄像头图标或选择扫描账户条形码等命令，然后使用设备的摄像头扫描 QR 代码。
   - 在 Set up device（设置设备）向导中，选择 Show secret key（显示私有密钥），然后在您的 MFA 应用程序中键入私有密钥。

   > **Important**
   >
   > 对二维码或私有配置密钥进行安全备份，或确保为您的账户启用多台 MFA 设备。您最多可以向 Amazon Web Services 账户根用户 和 IAM 用户注册 8 台当前支持的 MFA 类型任意组合的 MFA 设备。虚拟 MFA 设备可能变为不可用（例如，如果丢失了承载虚拟 MFA 设备的智能手机）。如果发生这种情况，并且您无法在没有其他 MFA 设备附加到用户的情况下登录账户，甚至无法通过 恢复根用户用户 MFA 设备 登录，则您将无法登录您的账户，您将不得不联系客户服务以删除对该账户的 MFA 保护。

   设备开始生成六位数编码。

7. 在向导的 MFA code 1（MFA 代码 1）框中，键入虚拟 MFA 设备上当前显示的一次性密码。请等候 30 秒，以便设备生成新的一次性密码。然后在 MFA code 2 (MFA 代码 2) 框中键入第二个一次性密码。选择 Add MFA（添加 MFA）。

   > **Important**
   >
   > 生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

为根用户启用硬件 TOTP 令牌（控制台）

您只能从 Amazon Web Services 管理控制台 为根用户配置和启用实体 MFA 设备，而不能从 Amazon CLI 或 Amazon API 配置和启用。

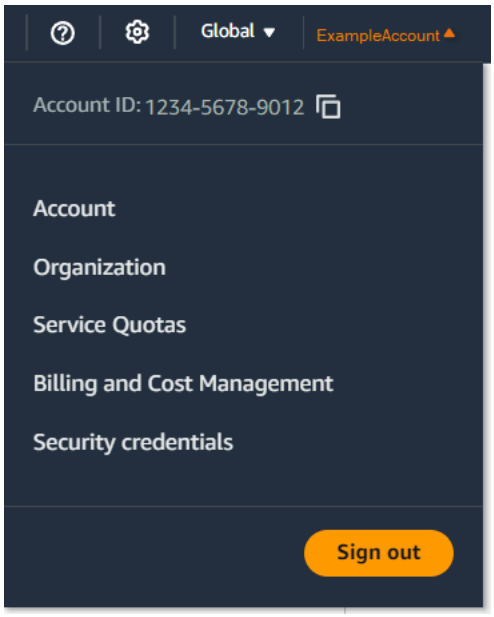
> **Note**
>
> 您可能会看到不同的文本，例如使用 MFA 登录和排除您的身份验证设备故障。不过，它们提供了相同的功能。在任一情况下，如果您无法使用替代身份验证因素验证您的账户电子邮件地址和电话号码，请与 Amazon Web Services 支持 联系以删除您的 MFA 设置。

为根用户启用硬件 TOTP 令牌（控制台）

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 用户指南》中的 以根用户身份登录 Amazon Web Services 管理控制台。

2. 在导航栏的右侧，选择您的账户名称，然后选择 Security Credentials（安全凭证）。



3. 展开多重身份验证 (MFA) 部分。
4. 选择分配 MFA 设备。

5. 在向导中，键入 Device name（设备名称），选择 Hardware TOTP token（硬件 TOTP 令牌），然后选择 Next（下一步）。
6. 在 Serial Number (序列号) 框中，键入在 MFA 设备背面找到的序列号。
7. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。

8. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。
9. 选择 Add MFA（添加 MFA）。MFA 设备现已与 Amazon Web Services 账户 相关联。

> **Important**
>
> 在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

下次使用根用户凭证登录时，您必须键入 MFA 设备生成的代码。

更改Amazon Web Services 账户根用户密码

您可以前往安全凭证或账户页面，更改电子邮件地址和密码。您还可以选择 Amazon 登录页面上的 Forgot password? 来重置您的密码。

要更改根用户密码，您必须以Amazon Web Services 账户根用户登录，而非以 IAM 用户身份登录。要了解如何重置已经忘记的根用户密码，请参阅 重置丢失或遗忘的根用户密码。

为保护您的密码，请务必遵循以下最佳实践：

- 定期更改密码。
- 妥善保管密码，因为任何知道您密码的人都可以访问您的账户。
- 不要为 Amazon 使用您在其他网站上使用的密码。
- 不要使用容易猜到的密码。此类密码包括 `secret`、`password`、`amazon`、`123456` 等。此外还应避免使用字典中的单词、姓名、电子邮件地址或可以轻松获取的其他个人信息。

> **Important**
>
> 使用 Amazon Organizations 管理的 Amazon Web Services 账户可能为成员账户启用了集中的根访问。这些成员账户没有根用户凭证，无法以根用户身份登录，并且无法恢复根用户密码。如果您需要执行需要根用户凭证的任务，则请联系您的管理员。

Amazon Web Services 管理控制台

为根用户更改密码

> **最小权限**
>
> 要执行下列步骤，您必须至少具有以下 IAM 权限：
>
> - 您必须以 Amazon Web Services 账户根用户身份登录，这将不需要其他 Amazon Identity and Access Management（IAM）权限。您无法以 IAM 用户或角色身份执行这些步骤。

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 User Guide》中的 Sign in to the Amazon Web Services 管理控制台 as the root user。
2. 在控制台的右上角选择您的账户名称或账号，然后选择安全凭证。
3. 在账户页面上的账户设置旁，选择编辑。出于安全考虑，系统会提示您重新验证身份。

   > **Note**
   >
   > 如果您没有看到编辑选项，则可能是因为您并非以账户根用户身份登录。以 IAM 用户或角色身份登录时，您无法修改账户设置。
4. 在更新账户设置页面的密码下，选择编辑。
5. 在更新密码页面上，填写当前密码、新密码和确认新密码字段。

> **Important**
>
> 请务必选择一个强密码。虽然您可能为 IAM 用户设置了账户密码策略，但该策略不适用于您的根用户。

Amazon 要求您的密码满足以下条件：

- 长度必须至少 8 个字符，最多 128 个字符。
- 必须至少包含以下字符类型中三种的组合：大写字母、小写字母、数字，以及 ! @ # $ % ^ & * () <> [] {} | _+-= 符号。
- 不得与您的 Amazon Web Services 账户名称或电子邮件地址相同。

6. 选择保存更改。

Amazon CLI or Amazon SDK

Amazon CLI 或来自任何一种 Amazon 的 API 操作均不支持此任务。您只能使用 Amazon Web Services 管理控制台执行此任务。

重置丢失或遗忘的根用户密码

当您首次创建 Amazon Web Services 账户 时，您提供了电子邮件地址和密码。这些是您的 Amazon Web Services 账户根用户凭证。如果您忘记根用户密码，可以从 Amazon Web Services 管理控制台重置密码。

使用 Amazon Organizations 管理的 Amazon Web Services 账户可能为成员账户启用了集中的根访问。这些成员账户没有根用户凭证，无法以根用户身份登录，并且无法恢复根用户密码。如果您需要执行需要根用户凭证的任务，则请联系您的管理员。

> **Important**
>
> 登录 Amazon 时遇到问题？ 请确保使用的是适合您用户类型的 Amazon 登录页面。如果您是 Amazon Web Services 账户根用户（账户拥有者），则可以使用您在创建 Amazon Web Services 账户 时设置的凭证登录 Amazon。如果您是 IAM 用户，则您的账户管理员可以向您提供用于登录 Amazon 的凭证。如果您需要请求支持，请不要使用此页面上的反馈链接，因为

该表由 Amazon 文档团队（而非 Amazon Web Services 支持）接收。请在联系我们页面上选择仍然无法登录 Amazon 账户，然后选择一个可用的支持选项。

重置根用户密码

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 User Guide》中的 Sign in to the Amazon Web Services 管理控制台 as the root user。

   > **Note**
   >
   > 如果您已使用 IAM 用户凭证登录到 Amazon Web Services 管理控制台，则必须注销，然后才能重置根用户密码。如果您看到特定于账户的 IAM 用户登录页面，请选择页面底部附近的使用根账户凭证登录。如有必要，请提供您的账户电子邮件地址并选择下一步来访问根用户登录页面。

2. 选择 Forgot your password?。

   > **Note**
   >
   > 如果您是 IAM 用户，则此选项不可用。忘记密码？选项仅适用于根用户账户。IAM 用户必须让其管理员重置忘记的密码。有关更多信息，请参阅我忘记了我 Amazon 账户的 IAM 用户密码。如果您通过 Amazon Web Services 访问门户登录，请参阅重置您的 IAM Identity Center 用户密码。

3. 提供与该账户关联的电子邮件地址。然后，提供 CAPTCHA 文本并选择 Continue。
4. 检查与您的 Amazon Web Services 账户 相关联的电子邮件邮箱中有无来自 Amazon Web Services 的邮件。该电子邮件来自以 `@verify.signin.aws` 结尾的地址。按照电子邮件中的指导进行操作。如果您在账户中未看到电子邮件，请检查垃圾邮件文件夹。如果您无法访问电子邮件，请参阅《Amazon 登录 用户指南》中的我无权访问我 Amazon 账户的电子邮件。

创建根用户的访问密钥

> **Warning**
>
> 强烈建议您不要创建根用户访问密钥对。由于只有少数任务需要根用户，而且您通常不经常执行这些任务，因此我们建议登录到 Amazon Web Services 管理控制台 来执行根用户任务。在创建访问密钥之前，请认真阅读长期访问密钥的替代方法。

您可以为根用户创建访问密钥，以便您可以在 Amazon Command Line Interface（Amazon CLI）中使用根用户凭证运行命令，或使用根用户凭证从某个 Amazon SDK 使用 API 操作，但我们不建议这样做。当您创建访问密钥时，您会将访问密钥 ID 和秘密访问密钥创建为一个集。在创建访问密钥的过程中，Amazon 仅允许您查看和下载一次访问密钥的秘密访问密钥部分。如果您未下载或丢失了访问密钥，则可删除访问密钥，然后创建新的访问密钥。您可以使用控制台、Amazon CLI 或 Amazon API 创建根用户访问密钥。

新创建的访问密钥的状态为已激活，这意味着，您可以使用访问密钥进行 CLI 和 API 调用。您最多可以为根用户分配两个访问密钥。

停用未使用的访问密钥。一旦访问密钥处于非活动状态，则无法将其用于 API 调用。非活动的密钥仍将计入您的限制。您随时可以创建或删除访问密钥。不过，当您删除访问密钥时，意味着永久删除且无法恢复。

Amazon Web Services 管理控制台

创建 Amazon Web Services 账户根用户的访问密钥

> **最小权限**
>
> 要执行下列步骤，您必须至少具有以下 IAM 权限：
>
> - 您必须以 Amazon Web Services 账户根用户身份登录，这将不需要其他 Amazon Identity and Access Management（IAM）权限。您无法以 IAM 用户或角色身份执行这些步骤。

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 User Guide》中的 Sign in to the Amazon Web Services 管理控制台 as the root user。

2. 在控制台的右上角选择您的账户名称或账号，然后选择安全凭证。
3. 在访问密钥部分，选择创建访问密钥。如果此选项不可用，则说明您拥有的访问密钥已达到最大数量。您必须首先删除一个现有的访问密钥，然后才能创建新的密钥。有关更多信息，请参阅 IAM 对象限额。
4. 在根用户访问密钥的替代方案页面上，查看相关安全建议。要继续操作，请选中该复选框，然后选择创建访问密钥。
5. 在检索访问密钥页面上，将显示您的访问密钥 ID。
6. 选择秘密访问密钥下的显示，然后从浏览器窗口复制访问密钥 ID 和私有密钥，并将其粘贴到其他位置。您还可以选择下载 .csv 文件，这时将会下载一个名为 `rootkey.csv` 的文件，其中包含访问密钥 ID 和私有密钥。将该文件安全保存在某个位置。
7. 选择完成。如果您不再需要该访问密钥，我们建议您将其删除，或者至少考虑将其停用，以免任何人不当使用。

Amazon CLI & SDKs

创建根用户的访问密钥

> **Note**
>
> 要以根用户身份运行以下命令或 API 操作，您必须事先拥有一个有效的访问密钥对。如果您没有任何访问密钥，请使用 Amazon Web Services 管理控制台 创建第一个访问密钥。然后可以将第一个访问密钥中的凭证与 Amazon CLI 结合使用，从而创建第二个访问密钥或删除访问密钥。

- Amazon CLI：aws iam create-access-key

  Example

```
$ aws iam create-access-key
{
    "AccessKey": {
        "UserName": "MyUserName",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "Status": "Active",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "CreateDate": "2021-04-08T19:30:16+00:00"
    }
```

```
}
```

- Amazon API：《IAM API 参考》中的 CreateAccessKey。

删除根用户的访问密钥

您不能使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 删除根用户访问密钥。

Amazon Web Services 管理控制台

删除根用户的访问密钥

> **最小权限**
>
> 要执行下列步骤，您必须至少具有以下 IAM 权限：
>
> - 您必须以 Amazon Web Services 账户根用户身份登录，这将不需要其他 Amazon Identity and Access Management（IAM）权限。您无法以 IAM 用户或角色身份执行这些步骤。

1. 打开 Amazon 管理控制台，使用根用户凭证登录。

   有关说明，请参阅《Amazon 登录 User Guide》中的 Sign in to the Amazon Web Services 管理控制台 as the root user。
2. 在控制台的右上角选择您的账户名称或账号，然后选择安全凭证。
3. 在访问密钥部分中，选择要删除的访问密钥，然后在操作下选择删除。

   > **Note**
   >
   > 您也可以停用访问密钥，而不是将其永久删除。这样，您将来可以继续使用它，而无需更改密钥 ID 或私有密钥。密钥处于非活动状态时，任何在 Amazon API 请求中使用该密钥的尝试都会失败，并返回访问被拒绝错误。
4. 在删除 <访问密钥 ID>对话框中，选择停用，输入访问密钥 ID 以确认要将其删除，然后选择删除。

Amazon CLI & SDKs

删除根用户的访问密钥

> **最小权限**
>
> 要执行下列步骤，您必须至少具有以下 IAM 权限：
>
> - 您必须以 Amazon Web Services 账户根用户身份登录，这将不需要其他 Amazon Identity and Access Management（IAM）权限。您无法以 IAM 用户或角色身份执行这些步骤。

- Amazon CLI：aws iam delete-access-key

  Example

  ```
  $ aws iam delete-access-key \
      --access-key-id AKIAIOSFODNN7EXAMPLE
  ```

  如果成功，此命令不会产生任何输出。
- Amazon API：DeleteAccessKey

需要根用户凭证的任务

我们建议您在 Amazon IAM Identity Center 中配置管理用户，以用来执行日常任务和访问 Amazon 资源。不过，您只能在以账户的根用户身份登录时才能执行下列任务。

为了简化在 Amazon Organizations 中跨成员账户管理特权根用户凭证的过程，您可以启用集中根访问来帮助您集中保护对您的 Amazon Web Services 账户高权限访问。集中管理成员账户的根访问权限 允许您集中删除和防止长期根用户凭证恢复，从而提高组织中的账户安全性。启用此功能后，您可以在成员账户上执行以下特权任务。

- 删除成员账户根用户凭证，以防止根用户的账户恢复。您还可以允许密码恢复操作，以恢复成员账户的根用户凭证。
- 删除一项配置错误的存储桶策略，此策略拒绝所有主体访问 Amazon S3 存储桶策略。
- 删除将会拒绝所有主体访问 Amazon SQS 队列的 Amazon Simple Queue Service 基于资源的策略。

账户管理任务

- 更改 Amazon Web Services 账户 设置。不属于 Amazon Organizations 的独立 Amazon Web Services 账户需要根凭证才能更新电子邮件地址、根用户密码和根用户访问密钥。其他账户设置（例如账户名称、联系人信息、备用联系人、付款货币偏好和 Amazon Web Services 区域）不需要根用户凭证。

  > **Note**
  >
  > 启用所有功能后，Amazon Organizations 可用于从管理账户和委托管理员账户集中管理成员账户设置。管理账户和委托管理员账户中的授权 IAM 用户或 IAM 角色可以关闭成员账户，并更新根电子邮件地址、账户名称、联系人信息、备用联系人和成员账户的 Amazon Web Services 区域。

- 关闭 Amazon Web Services 账户。不属于 Amazon Organizations 的独立 Amazon Web Services 账户需要根凭证才能关闭账户。借助 Amazon Organizations，您可以从管理账户和委托管理员账户集中关闭成员账户。
- 恢复 IAM 用户权限。如果唯一的 IAM 管理员意外撤消了自己的权限，您可以使用根用户身份登录来编辑策略并还原这些权限。

计费任务

- 激活 IAM 对账单和成本管理控制台的访问权限。
- 某些计费任务仅限于根用户。有关更多信息，请参阅《Amazon Billing 用户指南》中的管理 Amazon Web Services 账户。
- 查看特定税务发票。具有 aws-portal:ViewBilling 权限的 IAM 用户可以查看和下载 Amazon 欧洲的增值税发票，但不能查看和下载 Amazon Inc 或 Amazon Internet Services Private Limited（AISPL）的增值税发票。

Amazon GovCloud (US) 任务

- 注册 Amazon GovCloud (US)。
- 向 Amazon Web Services 支持 请求 Amazon GovCloud (US) 账户根用户访问密钥。

Amazon EC2 任务

- 已在预留实例 Marketplace 中注册为卖家。

Amazon KMS 任务

- 如果 Amazon Key Management Service 密钥变得无法管理，则管理员可以通过联系 Amazon Web Services 支持 来进行恢复；但是，Amazon Web Services 支持 会通过确认票证 OTP 来响应根用户的主电话号码进行授权。

Amazon Mechanical Turk 任务

- 将您的 Amazon Web Services 账户 关联到您的 mTurk 请求者账户。

Amazon Simple Storage Service 任务

- 配置 Amazon S3 存储桶以启用 MFA（多重身份验证）。
- 编辑或删除拒绝所有主体的 Amazon S3 存储桶策略。

  您可以使用特权操作来解锁存储桶策略配置错误的 Amazon S3 存储桶。有关更多信息，请参阅 在 Amazon Organizations 成员账户上执行特权任务。

Amazon Simple Queue Service 任务

- 编辑或删除拒绝所有主体的 Amazon SQS 基于资源的策略。

  您可以使用特权操作来解锁基于资源的策略配置错误的 Amazon SQS 队列。有关更多信息，请参阅 在 Amazon Organizations 成员账户上执行特权任务。

其他资源

有关 Amazon 根用户的更多信息，请参阅以下资源：

- 有关根用户问题的帮助，请参阅 排查根用户问题。
- 要在 Amazon Organizations 中集中管理根用户电子邮件地址，请参阅《Amazon Organizations 用户指南》中的更新成员账户的根用户电子邮件地址。

以下文章提供了有关使用根用户的更多信息。

- 哪些最佳实践有利于保护我的 Amazon Web Services 账户以及其中的资源？
- 如何创建 EventBridge 事件规则，已在我的根用户被人使用时通知我？
- 监控 Amazon Web Services 账户根用户活动并发出通知
- 监控 IAM 根用户活动

IAM 用户

> **Important**
>
> IAM 最佳实践建议您您要求人类用户使用与身份提供商的联合身份验证才能使用临时凭证访问 Amazon，而不是使用具有长期凭证的 IAM 用户。建议您仅在联合用户不支持的特定用例中使用 IAM 用户。

IAM 用户是您在 Amazon Web Services 账户 中创建的实体。IAM 用户表示使用 IAM 用户与 Amazon 互动的人类用户或工作负载。IAM 用户包括名称和凭证。

具备管理员权限的 IAM 用户与 Amazon Web Services 账户根用户 并不是一回事。有关根用户的更多信息，请参阅 Amazon Web Services 账户根用户。

Amazon 如何标识 IAM 用户

当您创建 IAM 用户时，IAM 提供以下这些方法来识别该用户：

- 该 IAM 用户的“易记名称”，这是您在创建 IAM 用户时指定的名称，如 `Richard` 或 `Anaya`。您将在 Amazon Web Services 管理控制台中看到这些名称。由于 IAM 用户名出现在 Amazon 资源名称 (ARN) 中，因此不建议在 IAM 名称中包含个人识别信息。有关 IAM 名称的要求和限制，请参阅 IAM 名称要求。
- IAM 用户的 Amazon 资源名称（ARN）。当您需要跨所有 Amazon 唯一标识 IAM 用户时，可以使用 ARN。例如，您可以使用 ARN 在 Amazon S3 存储桶的 IAM policy 中将 IAM 用户指定为 `Principal`。IAM 用户的 ARN 可能类似于以下内容：

  ```
  arn:aws:iam::account-ID-without-hyphens:user/Richard
  ```

- IAM 用户的唯一标识符。仅在您使用 API、Tools for Windows PowerShell 或 Amazon CLI 创建 IAM 用户时返回此 ID；控制台中不会显示此 ID。

有关这些标识符的更多信息，请参阅IAM 标识符。

IAM 用户和凭证

您可以通过不同方式访问 Amazon，具体取决于 IAM 用户凭证：

- 控制台密码：IAM 用户可键入该密码以登录交互式会话，例如 Amazon Web Services 管理控制台。禁用 IAM 用户的密码（控制台访问）可防止用户使用其登录凭证登录 Amazon Web Services 管理控制台。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。启用控制台访问权限的 IAM 用户也可以使用这些相同的凭证，通过 `aws login` Amazon CLI 命令对 Amazon CLI 和 SDK 访问进行身份验证。这些用户需要具有 SignInLocalDevelopmentAccess 权限。有关更多详细信息，请参阅《Amazon Command Line Interface 用户指南》中的 Amazon CLI 身份验证和访问凭证。
- 访问密钥：用于以编程方式调用 Amazon。但是，在为 IAM 用户创建访问密钥之前，还需要考虑更安全的替代方法。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 长期访问密钥的注意事项和替代方案。如果该 IAM 用户具有有效的访问密钥，它们将继续保持有效并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API，或 Amazon Console Mobile Application 进行访问。
- 与 CodeCommit 结合使用的 SSH 密钥：可用于向 CodeCommit 进行身份验证的采用 OpenSSH 格式的 SSH 公有密钥。
- 服务器证书：您可用于向某些 Amazon 服务进行身份验证的 SSL/TLS 证书。我们建议您使用 Amazon Certificate Manager (ACM) 来预置、管理和部署您的服务器证书。只有当您必须在 ACM 不支持的区域中支持 HTTPS 连接时，才应使用 IAM。要了解 ACM 支持的具体区域，请参阅《Amazon Web Services 一般参考》中的 Amazon Certificate Manager 端点和限额。

您可以选择最适合您的 IAM 用户的凭证。当您使用 Amazon Web Services 管理控制台 来创建 IAM 用户时，必须选择至少包含一个控制台密码或访问密钥。默认情况下，使用 Amazon CLI 或 Amazon API 创建的全新 IAM 用户没有任何类型的凭证。您必须根据使用案例为 IAM 用户创建凭证类型。

您可以使用以下选项来管理密码、访问密钥和多重身份验证（MFA）设备：

- 管理 IAM 用户的密码。创建和更改允许访问 Amazon Web Services 管理控制台的密码。设置密码策略以强制实施最小密码复杂性。允许 IAM 用户更改其密码。
- 管理 IAM 用户的访问密钥。创建和更新用于通过编程方式访问账户中的资源的访问密钥。
- 为 IAM 用户启用多重身份验证（MFA）。作为最佳实践，我们建议您要求账户中的所有 IAM 用户进行多重身份验证（MFA）。使用 MFA，IAM 用户必须提供两种形式的身份证明：首先，他们提供属于用户身份一部分的凭证（密码或访问密钥）。此外，他们还提供在硬件设备上或由智能手机或平板电脑上的应用程序生成的临时数字代码。

- 查找未使用的密码和访问密钥。拥有您账户或您账户中的 IAM 用户的密码或访问密钥的任何人都可以访问您的 Amazon 资源。为保障安全，最佳实践是在 IAM 用户不再需要密码和访问密钥时将其删除。
- 下载您账户的凭证报告。您可以生成和下载列出您账户中所有 IAM 用户及其各个凭证状态（包括密码、访问密钥和 MFA 设备）的凭证报告。对于密码和访问密钥，凭证报告将显示多久前使用了密码或访问密钥。

IAM 用户和权限

默认情况下，全新的 IAM 用户没有执行任何操作的权限。他们无权执行任何 Amazon 操作或访问任何 Amazon 资源。采用单独 IAM 用户的优势在于可单独向每个用户分配权限。您可以向几个用户分配管理权限，而这些用户随后可管理您的 Amazon 资源，甚至创建和管理其他 IAM 用户。但在大多数情况下，您希望限制用户的权限，使其只能访问工作所需的任务（Amazon 操作）和资源。

设想一个名为 Diego 的用户。当您创建 IAM 用户 `Diego` 时，您可以为其创建密码并附加权限，以使其能够启动特定 Amazon EC2 实例以及从 Amazon RDS 数据库中的表读取 (GET) 信息。有关如何创建 IAM 用户并向其授予初始凭证和权限的过程，请参阅在 Amazon Web Services 账户 中创建 IAM 用户。有关如何更改现有用户的权限的过程，请参阅更改 IAM 用户的权限。有关如何更改用户的密码或访问密钥的过程，请参阅Amazon 中的用户密码和管理 IAM 用户的访问密钥。。

您还可以向您的 IAM 用户添加权限边界。权限边界是一项高级功能，可让您使用 Amazon 托管策略来限制基于身份的策略可向 IAM 用户或角色授予的最大权限。有关策略类型和用法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

IAM 用户和账户

每个 IAM 用户均与一个 Amazon Web Services 账户（且仅一个）关联。由于 IAM 用户是在您的 Amazon Web Services 账户 中定义的，因此不需要向 Amazon 报备付款方式。IAM 用户在您的账户中执行的任何 Amazon 活动产生的费用均计入您的账户。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

作为服务账户的 IAM 用户

IAM 用户是 IAM 中具有相关凭证和权限的资源。IAM 用户可以表示一个人或使用此人的凭证向 Amazon 提出请求的应用程序。这通常被称为服务账户。如果您选择在应用程序中使用 IAM 用户的长期凭证，请勿直接将访问密钥嵌入您的应用程序代码。使用 Amazon 开发工具包和 Amazon

Command Line Interface，可以在已知位置放置访问密钥，这样就不必将其保留在代码中。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 正确管理 IAM 用户访问密钥。另外，作为最佳实践，您可以使用临时安全凭证（IAM 角色）代替长期访问密钥。

IAM 用户如何登录 Amazon

要以 IAM 用户的身份登录到 Amazon Web Services 管理控制台，除了您的用户名和密码外，您还须提供您的账户 ID 或账户别名。当您的管理员在控制台中创建您的 IAM 用户时，他们应该已经向您发送了登录凭证，其中包括您的用户名和账户登录页面的 URL，内含您的账户 ID 或账户别名。

```
https://My_AWS_Account_ID.signin.aws.amazon.com/console/
```

> **提示**
>
> 要在 Web 浏览器中为您的账户登录页面创建书签，您应在标签条目中手动键入您的账户的登录 URL。不要使用 Web 浏览器的书签功能，因为重定向会掩盖登录 URL。

您也可以在以下常规登录端点登录，然后手动键入您的账户 ID 或账户别名：

```
https://console.aws.amazon.com/
```

为方便起见，Amazon 登录页面使用浏览器 Cookie 记住 IAM 用户名和账户信息。下次用户转到 Amazon Web Services 管理控制台 中的任何页面时，控制台会使用 cookie 将用户重定向到账户登录页面。

您只能访问您的管理员在附加到您的 IAM 用户身份的策略中指定的 Amazon 资源。要在控制台开展工作，您必须有权限执行控制台执行的操作（例如列出和创建 Amazon 资源）。有关更多信息，请参阅 适用于 Amazon 资源的 Access Management 和 IAM 基于身份的策略示例。

> **Note**
>
> 如果贵企业现在有一个身份系统，您可能需要创建单点登录 (SSO) 选项。SSO 向用户提供对您账户 Amazon Web Services 管理控制台 的访问权限，而不要求他们具有 IAM 用户身份。SSO 也无需用户单独登录您的组织的网站和 Amazon。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。

在 CloudTrail 中记录登录详细信息

如果您允许 CloudTrail 将登录事件记录到您的日志中，您需要了解 CloudTrail 如何选择在何处记录事件。

- 如果您的用户直接登录到控制台，则系统会根据所选服务控制台是否支持区域，将他们重定向到全局或区域登录端点。例如，主控制台主页支持区域，因此，如果您登录以下 URL：

  ```
  https://alias.signin.aws.amazon.com/console
  ```

  您会被重定向到 `https://us-east-2.signin.aws.amazon.com` 这样的区域登录端点，使得该区域日志中产生一个区域 CloudTrail 日志条目：

  另一方面，Amazon S3 控制台不支持区域，因此，如果您登录到以下 URL

  ```
  https://alias.signin.aws.amazon.com/console/s3
  ```

  Amazon 会将您重定向到全局登录端点 `https://signin.aws.amazon.com`，从而产生一个全局 CloudTrail 日志条目。
- 您可以通过使用类似如下的 URL 语法登录到启用区域的主控制台主页，来手动请求特定区域网站端点：

  ```
  https://alias.signin.aws.amazon.com/console?region=ap-southeast-1
  ```

  Amazon 将您重定向到 `ap-southeast-1` 区域登录端点并使该区域中产生一个 CloudTrail 日志事件。

有关 CloudTrail 和 IAM 的更多信息，请参阅使用 CloudTrail 记录 IAM 事件。

如果用户需要编程式访问来使用您的账户，则可以为每位用户创建访问密钥对（访问密钥 ID 和秘密访问密钥）。但是，在为用户创建访问密钥之前，还需要考虑更安全的替代方法。有关更多信息，请参阅《Amazon Web Services 一般参考》中的 长期访问密钥的注意事项和替代方案。

其他资源

以下资源可帮助您了解有关 Amazon 登录的更多信息。

- 《Amazon 登录用户指南》https://docs.amazonaws.cn/signin/latest/userguide/what-is-sign-in.html本指南可帮助您了解不同类型的用户登录 Amazon Web Services（Amazon）的不同方式。

- 在 Amazon Web Services 管理控制台中的单个 Web 浏览器中，您最多可以同时登录五个不同的身份。有关详细信息，请参阅《Amazon Web Services 管理控制台入门指南》中的登录多个账户。

已启用 MFA 的登录

配置了多重身份验证（MFA）设备的用户必须使用其 MFA 设备登录 Amazon Web Services 管理控制台。在用户输入其登录凭证后，Amazon 将检查用户的账户以查看该用户是否需要 MFA。

> **Important**
>
> 如果您通过 Amazon STS `GetFederationToken` API 调用并使用访问密钥和私有密钥凭证直接访问 Amazon Web Services 管理控制台，则不需要 MFA。有关更多信息，请参阅 使用访问密钥和私有密钥凭证访问控制台。

以下主题介绍了用户在需要使用 MFA 时如何完成登录。

主题

- 多个已启用 MFA 的设备
- FIDO 安全密钥
- 虚拟 MFA 设备
- 硬件 TOTP 令牌

多个已启用 MFA 的设备

如果用户以 Amazon Web Services 账户 根用户或 IAM 用户的身份登录 Amazon Web Services 管理控制台，且为该账户启用了多个 MFA 设备，则他们只需要使用一台 MFA 设备即可登录。用户使用用户密码进行身份验证后，他们可以选择要使用哪种 MFA 设备类型来完成身份验证。然后，系统会提示用户使用他们选择的设备类型来进行身份验证。

FIDO 安全密钥

如果 MFA 是用户必须使用的，则会显示另一个登录页面。用户需要点击 FIDO 安全密钥。

> **Note**
>
> Google Chrome 用户不应选择弹出窗口中的任何要求 Verify your identity with amazon.com（通过 amazon.com 验证您的身份）的可用选项。您只需要点击安全密钥即可。

与其他 MFA 设备不同，FIDO 安全密钥不同步。如果 FIDO 安全密钥丢失或损坏，管理员可以停用它。有关更多信息，请参阅 停用 MFA 设备（控制台）。

有关支持 Amazon 所支持的 WebAuthn 和 FIDO 合规设备的浏览器的信息，请参阅 使用密钥或安全密钥的受支持配置。

虚拟 MFA 设备

如果 MFA 是用户必须使用的，则会显示另一个登录页面。在 MFA code (MFA 代码) 框中，用户必须输入 MFA 应用程序提供的数字代码。

如果 MFA 代码正确，则用户可以访问 Amazon Web Services 管理控制台。如果代码不正确，则用户可以使用其他代码重试。

虚拟 MFA 设备可能会不同步。如果用户尝试多次后都无法登录 Amazon Web Services 管理控制台，系统将提示用户同步虚拟 MFA 设备。用户可以按照屏幕上的提示同步虚拟 MFA 设备。有关如何在您的 Amazon Web Services 账户 中同步代表用户的设备的信息，请参阅 重新同步虚拟和硬件 MFA 设备。

硬件 TOTP 令牌

如果 MFA 是用户必须使用的，则会显示另一个登录页面。在 MFA code（MFA 代码）框中，用户必须输入硬件 TOTP 令牌提供的数字代码。

如果 MFA 代码正确，则用户可以访问 Amazon Web Services 管理控制台。如果代码不正确，则用户可以使用其他代码重试。

硬件 TOTP 令牌可能会不同步。如果用户尝试多次后都无法登录 Amazon Web Services 管理控制台，系统将提示用户同步 MFA 令牌设备。用户可以根据屏幕上的提示同步 MFA 令牌设备。有关如何在您的 Amazon Web Services 账户 中同步代表用户的设备的信息，请参阅 重新同步虚拟和硬件 MFA 设备。

在 Amazon Web Services 账户 中创建 IAM 用户

> **Important**
>
> IAM 最佳实践建议您您要求人类用户使用与身份提供商的联合身份验证才能使用临时凭证访问 Amazon，而不是使用具有长期凭证的 IAM 用户。我们建议您仅在联合用户不支持的特定用例中使用 IAM 用户。

创建 IAM 用户并使该用户能够执行任务的过程包含以下步骤：

1. 在 Amazon Web Services 管理控制台、Amazon CLI、Tools for Windows PowerShell 中或使用 Amazon API 操作创建用户。如果您在 Amazon Web Services 管理控制台 中创建用户，则将根据您的选择自动处理步骤 1 到步骤 4。如果以编程方式创建 IAM 用户，则必须分别执行上述每个步骤。
2. 根据用户所需的访问类型为用户创建凭证：
   - 启用控制台访问 – 可选：如果用户需要访问 Amazon Web Services 管理控制台，请为用户创建密码。禁用用户的控制台访问可防止用户使用其用户名和密码登录 Amazon Web Services 管理控制台。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。

     > **Tip**
     >
     > 请仅创建用户需要的凭证。例如，对于仅需要通过 Amazon Web Services 管理控制台进行访问的用户，请勿创建访问密钥。
3. 给予用户执行所需任务的权限。建议将 IAM 用户放入组内，通过附加到这些组的策略来管理权限。但是，您还可以通过将权限策略直接附加到用户来授予权限。如果使用控制台添加用户，则您可以将现有用户的权限复制到新用户。

   您还可以指定定义用户可以拥有的最大权限的策略，由此添加权限边界来限制用户的权限。权限边界不授予任何权限。

   有关创建用于授予权限或设置权限边界的自定义权限策略的说明，请参阅 使用客户管理型策略定义自定义 IAM 权限。
4. （可选）通过附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
5. 向用户提供必要的登录信息。信息包括密码以及用户在其中提供这些凭证的账户登录页面的控制台 URL。有关更多信息，请参阅 IAM 用户如何登录 Amazon。

6. (可选) 为用户配置多重验证 (MFA)。MFA 要求用户在每次登录 Amazon Web Services 管理控制台时都提供一次性的代码。
7. （可选）向 IAM 用户授予管理自己安全凭证所需的权限。（默认状态下，IAM 用户没有权限管理自己的凭证。）有关更多信息，请参阅 允许 IAM 用户更改自己的密码。

> **Note**
>
> 如果您使用控制台创建用户并选择用户必须在下次登录时创建新密码（推荐），则用户具有所需的权限。

有关创建用户时需要的权限的信息，请参阅访问 IAM 资源所需的权限。

有关为特定应用场景创建 IAM 用户的说明，请参阅以下主题：

- 创建用于紧急访问的 IAM 用户
- 为无法使用 IAM 角色的工作负载创建 IAM 用户

查看 IAM 用户

您可以列出 Amazon Web Services 账户 中或特定 IAM 组中的 IAM 用户，也可以列出用户所属的所有 IAM 组。有关为列出用户而需要的权限的信息，请参阅访问 IAM 资源所需的权限。

要列出账户中的所有 IAM 用户

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。

控制台会显示 Amazon Web Services 账户 中的 IAM 用户。

Amazon CLI

运行以下命令：

- `aws iam list-users`

API

调用以下操作：

- ListUsers

要列出 IAM 群组中的 IAM 用户

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 User groups（用户组）。
4. 选择用户组的名称。

Users（用户）选项卡上列出了属于该群组的 IAM 用户。

Amazon CLI

运行以下命令：

- aws iam get-group

API

调用以下操作：

- GetGroup

列出用户所属的所有 IAM 组

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择 IAM 用户的名称。

5. 选择 Groups（群组）选项卡，以显示包含当前用户的群组的列表。

Amazon CLI

运行以下命令：

- `aws iam list-groups-for-user`

API

调用以下操作：

- `ListGroupsForUser`

后续步骤

获得 IAM 用户列表后，可以按照以下步骤重命名、删除或停用 IAM 用户。

- 重命名 IAM 用户
- 移除或停用 IAM 用户

重命名 IAM 用户

> **Note**
>
> 作为最佳实践，我们建议您要求人类用户使用带有身份提供程序的联合身份验证才能使用临时凭证访问 Amazon。如果您遵循最佳实践，则无法管理 IAM 用户和组。相反，您的用户和组是在 Amazon 外部进行管理的，并且能够以联合身份访问 Amazon 资源。联合身份是来自企业用户目录、Web 身份提供程序、Amazon Identity Service 的用户，或任何使用通过身份源提供的凭证来访问 Amazon 服务的用户。联合身份使用其身份提供商定义的组。如果您使用的是 Amazon IAM Identity Center，请参阅《Amazon IAM Identity Center 用户指南》中的管理 IAM Identity Center 中的身份，了解有关在 IAM Identity Center 中创建用户和组的信息。

亚马逊云科技提供多种工具来管理 Amazon Web Services 账户 中的 IAM 用户。您可以列出账户中或用户组中的 IAM 用户，也可以列出用户所属的所有 IAM 组。您可以重命名或更改 IAM 用户的路径。如果您使用的是联合身份而不是 IAM 用户，则可以从 Amazon 账户中删除 IAM 用户，或停用该用户。

有关添加、更改或删除 IAM 用户的托管策略的更多信息，请参阅 更改 IAM 用户的权限。有关为 IAM 用户管理内联策略的信息，请参阅 添加和删除 IAM 身份权限、编辑 IAM 策略 和 删除 IAM 策略。作为最佳实践，请使用托管式策略而不是内联策略。Amazon 托管式策略可用于为很多常用案例提供权限。请记住，Amazon 托管策略可能不会为您的特定使用场景授予最低权限许可，因为它们可供所有 Amazon 客户使用。因此，我们建议通过定义特定于您的使用场景的客户管理型策略来减少许可。有关更多信息，请参阅 Amazon托管策略。有关专为特定任务函数制定的 Amazon 托管策略的更多信息，请参阅 工作职能的 Amazon 托管策略。

要了解有关验证 IAM policy 的更多信息，请参阅 IAM 策略验证。

> **Tip**
>
> IAM Access Analyzer 可以分析您的 IAM 角色使用的服务和操作，然后生成您可以使用的精细策略。测试每个生成的策略后，可以将该策略部署到生产环境中。这可确保您仅向工作负载授予所需的权限。有关策略生成的更多信息，请参阅 IAM Access Analyzer 策略生成。

有关管理 IAM 用户密码的信息，请参阅 管理 IAM 用户的密码。

重命名 IAM 用户

要更改用户的名称或路径，必须使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API。控制台中没有用于重命名用户的选项。有关为将用户重命名而需要的权限的信息，请参阅访问 IAM 资源所需的权限。

当您更改用户名或路径时，发生以下情况：

- 应用于用户的所有策略采用新用户名继续生效。
- 采用新用户名的用户保留在原来的 IAM 组。
- 用户的唯一 ID 保持不变。有关唯一 ID 的更多信息，请参阅 唯一标识符。
- 任何将该用户视为主体（向该用户授予访问权限）的资源或角色策略均自动更新以使用新用户名或路径。例如，Amazon SQS 中任何基于队列的策略或 Amazon S3 中任何基于资源的策略都会自动更新，以使用新名称和路径。

IAM 不自动更新将该用户视为资源的策略以使用新用户名或路径；必须由您手动更新。例如，假设向用户 `Richard` 附加了一个策略，该策略使该用户可管理其自己的安全凭证。如果管理员将 `Richard` 重命名为 `Rich`，则管理员还需要更新该策略以将资源从此：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Richard
```

改为此：

```
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Rich
```

如果管理员更改路径，则也会发生这种情况；管理员需要更新策略以反映该用户使用新路径。

重命名用户

- Amazon CLI：aws iam update-user
- Amazon API：UpdateUser

移除或停用 IAM 用户

最佳实操建议您从 Amazon Web Services 账户 中移除未使用的 IAM 用户。如果您想保留 IAM 用户的凭证以供将来使用，可以停用用户的访问权限，而不是从账户中删除凭证。有关更多信息，请参阅 停用 IAM 用户。

> **Warning**
>
> IAM 用户及其访问密钥一经删除，便无法恢复。

先决条件 – 查看 IAM 用户访问权限

在移除用户之前，请查看其最近的服务级活动。这有助于防止删除正在使用访问权限的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

移除 IAM 用户（控制台）

使用 Amazon Web Services 管理控制台 来移除 IAM 用户时，IAM 将自动删除以下关联信息：

- IAM 用户标识符
- 任何群组成员资格 - 也就是说，该 IAM 用户将从其所属的任何 群组中移除
- 任何与该 IAM 用户关联的密码

- 嵌入到该 IAM 用户中的所有内联策略（通过用户组权限应用于用户的策略不受影响）

> **Note**
>
> 当您删除用户时，IAM 会移除附加到 IAM 用户的任何托管策略，但不会删除托管策略。

- 任何关联的 MFA 设备

要移除 IAM 用户（控制台）

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择用户，然后选中要删除的 IAM 用户名旁边的复选框。
4. 在页面的顶部，选择 Delete（移除）。

   > **Note**
   >
   > 如果任何用户有激活的访问密钥，则必须在删除用户之前停用访问密钥。有关更多信息，请参阅 要停用 IAM 用户的访问密钥。

5. 在确认对话框的文本输入字段中输入用户名，以确认移除用户。选择 Delete（移除）。

控制台显示状态消息，通知 IAM 用户已被移除。

移除 IAM 用户 (Amazon CLI)

与 Amazon Web Services 管理控制台 不同，在使用 Amazon CLI 移除 IAM 用户时，您必须手动删除附加到该 IAM 用户的项目。此步骤演示了该过程。

从您的 Amazon Web Services 账户 (Amazon CLI) 中移除 IAM 用户

1. 删除用户的密码（如果用户有）。

   ```
   aws iam delete-login-profile
   ```

2. 删除用户的访问密钥（如果用户有）。

aws iam list-access-keys（用于列出用户的访问密钥）和 aws iam delete-access-key

3. 删除用户的签名证书。注意，当您删除安全凭证时，凭证永远消失，无法恢复。

   aws iam list-signing-certificates（用于列出用户的签名证书）和 aws iam delete-signing-certificate

4. 请删除用户的 SSH 公有密钥（如果用户具有）。

   aws iam list-ssh-public-keys（用于列出用户的 SSH 公有密钥）和 aws iam delete-ssh-public-key

5. 请删除用户的 Git 凭证。

   aws iam list-service-specific-credentials（用于列出用户的 Git 凭证）和 aws iam delete-service-specific-credential

6. 停用用户的多重身份验证 (MFA) 设备（如果用户具有）。

   aws iam list-mfa-devices（用于列出用户的 MFA 设备）、aws iam deactivate-mfa-device（用于停用设备）和 aws iam delete-virtual-mfa-device（用于永久删除虚拟 MFA 设备）

7. 请删除用户的内联策略。

   aws iam list-user-policies（用于列出用户的内联策略）和 aws iam delete-user-policy（用于删除策略）

8. 分离附加到用户的任何托管策略。

   aws iam list-attached-user-policies（用于列出附加到用户的托管策略）和 aws iam detach-user-policy（用于分离策略）

9. 从任何 IAM 组中删除用户。

   aws iam list-groups-for-user（用于列出用户所属的 IAM 组）和 aws iam remove-user-from-group

10. 删除用户。

    aws iam delete-user

停用 IAM 用户

如果 IAM 用户暂时退出贵公司，您可能需要将其停用。您可以保留其 IAM 用户凭证，但仍然阻止其进行 Amazon 访问。

要停用用户，请创建并附加策略以拒绝该用户访问 Amazon。您可以稍后恢复该用户的访问权限。

以下是您可以附加到用户以拒绝其访问的拒绝策略的两个示例。

以下策略不包含时间限制。您必须删除该策略才能恢复用户的访问权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*"
      }
   ]
}
```

以下策略包含一个条件，即策略自 2024 年 12 月 24 日晚上 11:59（UTC）起生效，并于 2025 年 2 月 28 日晚上 11:59（UTC）终止。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
          "DateGreaterThan": {"aws:CurrentTime": "2024-12-24T23:59:59Z"},
          "DateLessThan": {"aws:CurrentTime": "2025-02-28T23:59:59Z"}
```

```
            }
        }
    ]
}
```

控制 IAM 用户对 Amazon Web Services 管理控制台 的访问

具有权限的 IAM 用户通过 Amazon Web Services 管理控制台 登录到 Amazon Web Services 账户 可以访问您的 Amazon 资源。以下列表显示了您可授权 IAM 用户通过 Amazon Web Services 管理控制台 访问您的 Amazon Web Services 账户 资源的几种方式。同时，还列出了 IAM 用户可通过 Amazon 网站访问其他 Amazon 账户功能的方式。

> **Note**
>
> 使用 IAM 不会产生任何费用。

这些区域有：Amazon Web Services 管理控制台

> 您为需要访问 Amazon Web Services 管理控制台 的每位 IAM 用户创建一个密码。用户通过启用了 IAM 的 Amazon Web Services 账户 登录页面访问该控制台。有关访问登录页面的更多信息，请参阅《Amazon 登录 用户指南》中的如何登录 Amazon。有关创建密码的信息，请参阅 Amazon 中的用户密码。
>
> 您可以通过删除 IAM 用户的密码来阻止其访问 Amazon Web Services 管理控制台。这样可以阻止他们使用其登录凭证登录 Amazon Web Services 管理控制台。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API，或 Amazon Console Mobile Application 进行访问。

您的 Amazon 资源（例如 Amazon EC2 实例、Amazon S3 存储桶等）

> 即使您的 IAM 用户持有密码，也仍需拥有访问您的 Amazon 资源的权限。当您创建 IAM 用户时，该用户默认没有任何权限。若要为 IAM 用户授予所需的权限，您需要为其附加策略。如果您有许多使用相同资源来执行相同任务的 IAM 用户，则可将这些 IAM 用户分配至一个组。然后将权限分配给该组。有关创建 IAM 用户和组的信息，请参阅 IAM 身份。有关使用策略来设置许可的信息，请参阅 适用于 Amazon 资源的 Access Management。

Amazon 开发论坛

任何人均可阅读 Amazon 开发论坛上的帖子。希望在 Amazon 开发论坛上发布问题或评论的用户可使用他们的用户名进行操作。当用户初次在 Amazon 论坛上发帖时，将提示该用户输入昵称和电子邮件地址。只有该用户可以在 Amazon 论坛中使用该昵称

您的 Amazon Web Services 账户 账单和使用情况信息

您可授权用户访问您的 Amazon Web Services 账户 账单和使用情况信息。有关更多信息，请参阅 Amazon Billing 用户指南中的控制对账单信息的访问权限。

您的 Amazon Web Services 账户 个人资料信息

用户无法访问您的 Amazon Web Services 账户 个人资料信息。

您的 Amazon Web Services 账户 安全凭证

用户无法访问您的 Amazon Web Services 账户 安全凭证。

> **Note**
>
> IAM policy 控制访问不受接口限制。例如，您可以为用户提供密码以访问 Amazon Web Services 管理控制台。该用户（或该用户所属的任何组）的策略将控制用户可在 Amazon Web Services 管理控制台 中执行的操作。或者，您可以为用户提供 Amazon 访问密钥以对 Amazon 进行 API 调用。这些策略将控制用户可以通过使用这些访问密钥进行身份验证的数据库或客户端可以调用哪些操作。

更改 IAM 用户的权限

您可以更改您 Amazon Web Services 账户 中的 IAM 用户的权限，方法是更改用户的组成员资格、复制现有用户的权限、直接为用户附加策略或设置权限边界。权限边界控制用户可以具有的最大权限。权限边界是一项高级 Amazon 功能。

有关您修改用户的权限所需的权限的信息，请参阅访问 IAM 资源所需的权限。

主题

- 查看用户访问
- 基于用户的访问活动生成策略
- 向用户添加权限（控制台）

- 更改用户的权限（控制台）
- 要从用户删除权限策略（控制台）
- 要从用户删除权限边界（控制台）
- 添加和删除用户的权限（Amazon CLI 或 Amazon API）

查看用户访问

在更改用户的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

基于用户的访问活动生成策略

有时，您可能会向 IAM 实体（用户或角色）授予超出其需要的权限。为帮助您优化授予的权限，您可以根据实体的访问活动生成 IAM policy。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含实体在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的托管策略，然后将其附加到 IAM 实体。这样，您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。要了解更多信息，请参阅 IAM Acess Analyzer 策略生成。

向用户添加权限（控制台）

IAM 提供了三种方法来向用户添加权限策略：

- 将 IAM 用户添加到组 - 使用户成为一个组的成员。来自该组的策略将附加到用户。
- 复制现有 IAM 用户的权限 - 复制所有组成员资格、附加的托管策略、内联策略以及源用户的任何现有权限边界。
- 直接为 IAM 用户附加策略 - 直接为用户附加托管策略。为方便管理权限，请将策略附加到组，然后让 IAM 用户成为相应组的成员。

> **Important**
>
> 如果用户具有权限边界，则您为用户添加的权限不能超过权限边界所允许的权限。

要通过将 IAM 用户添加到组来添加权限

将 IAM 用户添加到 IAM 组后，该用户的权限会立即更新为该组所定义的权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择 IAM 用户的名称。
5. 选择 Groups（群组）选项卡，以显示包含当前用户的群组的列表。
6. 选择 Add user to groups（将用户添加到群组）。
7. 选中您希望用户加入的每个组对应的复选框。列表显示了每个组的名称以及用户在成为该组成员后获得的策略。
8. （可选）您可以选择 Create group（创建群组）来定义新的群组。如果您想将用户添加到与现有群组具有不同附加策略的群组中，这将非常有用：

   a. 在新选项卡中，对于 User group name（用户组名），键入您的新组名称。

   > **Note**
   >
   > Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。组名称可以是最多由 128 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您不能创建名为 TESTGROUP 和 testgroup 的两个组。

   b. 选中您要附加到组的托管式策略对应的一个或多个复选框。您还可以通过选择 Create policy 来创建新的托管策略。如果您这样做，请在创建新的策略之后返回到此浏览器选项卡或窗口；选择 Refresh，然后选择新策略以将其附加到您的组。有关更多信息，请参阅使用客户管理型策略定义自定义 IAM 权限。

   c. 选择创建用户组。

   d. 返回到原始选项卡，刷新您的组列表。然后选中您的新组对应的复选框。

9. 选择 Add user to group(s)（将用户添加到群组）。

控制台会显示一条状态消息，通知您用户已添加到您指定的群组中。

要通过从其他 IAM 用户复制来添加权限

如果您选择通过复制权限来为 IAM 用户添加权限，那么 IAM 会复制指定用户的所有组成员资格、附加托管策略、内联策略和任何现有权限边界，并立即将其应用到当前选定的用户。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择 IAM 用户的名称。
5. 在 Permissions（权限）选项卡中，选择 Add permissions（添加权限）。
6. 在 Add permissions（添加权限）页面上，选择 Copy permissions（复制权限）。列表显示了可用 IAM 用户以及其组成员资格和附加的策略。
7. 选择您想要复制其权限的用户旁边的单选按钮。
8. 选择 Next（下一步），以查看要对用户所做的更改列表。然后选择 Add permissions（添加权限）。

控制台会显示一条状态消息，通知您权限已从您指定的 IAM 用户那里复制。

要通过直接将策略附加到 IAM 用户来添加权限

您可以直接将托管策略附加到 IAM 用户。更新后的权限会立即应用。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择 IAM 用户的名称。
5. 在 Permissions（权限）选项卡中，选择 Add permissions（添加权限）。
6. 在 Add permissions（添加权限）页面上，选择 Attach policies directly（直接附加策略）。Permissions policies（权限策略）列表显示可用策略及其策略类型和附加实体。
7. 选择要附加的 Policy name（策略名称）旁边的单选按钮。

8. 选择 Next（下一步），以查看要对用户所做的更改列表。然后选择 Add permissions（添加权限）。

   控制台会显示一条状态消息，通知您已将策略添加到指定的 IAM 用户。

要为 IAM 用户设置权限边界

权限边界是 Amazon 中管理权限的高级功能，用于设置 IAM 用户可拥有的最大权限。设置权限边界会立即将 IAM 用户的权限限制在边界内，而与授予的其他权限无关。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择要更改其权限边界的 IAM 用户的名称。
5. 选择 Permissions（权限）选项卡。如有必要，打开 Permissions boundary （权限边界）部分，然后选择 Set permissions boundary（设置权限边界）。
6. 在 Set permissions boundary（设置权限边界）页面中的 Permissions policies（权限策略）下，选择您要用于权限边界的策略。
7. 选择 Set boundary（设置边界）。

   控制台会显示一条状态消息，通知您权限边界已添加。

更改用户的权限（控制台）

IAM 允许您通过以下方式更改与用户关联的权限：

- 编辑权限策略 - 编辑用户的内联策略、用户的组的内联策略，或编辑直接附加到用户或从组附加到用户的托管策略。如果用户具有权限边界，则您提供的权限不能超过用作用户的权限边界的策略所允许的权限。
- 更改权限边界 - 更改用作用户的权限边界的策略。这可以扩大或限制用户可以具有的最大权限。

编辑附加到用户的权限策略

更改权限会立即更新用户的访问权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择要更改其权限边界的 IAM 用户的名称。
5. 选择 Permissions（权限）选项卡。如有必要，打开 Permissions boundary（权限边界）部分。
6. 请选择要编辑的策略的名称以查看有关该策略的详细信息。选择 Entities attached（附加的实体）选项卡，查看编辑策略后可能受影响的其他实体（IAM 用户、组和角色）。
7. 选择 Permissions（权限）选项卡并查看策略授予的权限。要更改权限，请选择 Edit（编辑）。
8. 编辑该策略并解决任何策略验证建议。有关更多信息，请参阅 编辑 IAM 策略。
9. 选择 Next（下一步），查看策略摘要，然后选择 Save changes（保存更改）。

控制台会显示一条状态消息，通知您策略已更新。

要为用户更改权限边界

更改权限边界会立即更新用户的访问权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择要更改其权限边界的 IAM 用户的名称。
5. 选择 Permissions（权限）选项卡。如有必要，打开 Permissions boundary（权限边界）部分，然后选择 Change boundary（更改边界）。
6. 选择要用于权限边界的策略。
7. 选择 Set boundary（设置边界）。

控制台会显示一条状态消息，通知您权限边界已更改。

要从用户删除权限策略（控制台）

删除权限策略会立即更新用户的访问权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 选择要删除其权限策略的用户的名称。
5. 选择权限选项卡。
6. 如果要通过移除现有策略来撤消权限，请查看 Attached via（附加方式）来了解用户如何获取该策略，然后再选择 Remove（删除）以删除策略：
   - 如果该策略因组成员资格而适用，则选择 Remove（删除），从组中删除该用户。请记住，可向单个组中附加多个策略。如果您从组中删除某个用户，该用户将失去对其通过组成员资格接收的所有策略的访问权限。
   - 如果该策略是直接附加到该用户的托管策略，则选择 Remove（删除），从该用户中分离该策略。这不会影响该策略本身或该策略可能附加到的任何其他实体。
   - 如果该策略是内联嵌入式策略，则选择 Remove（删除），从 IAM 中删除该策略。直接附加到用户的内联策略只能在该用户中存在。

   如果策略是通过组成员资格授予用户的，则控制台会显示一条状态消息，通知您该 IAM 用户已从 IAM 组移除。如果策略是直接附加或内联的，则状态消息会通知您该策略已被删除。

要从用户删除权限边界（控制台）

删除权限边界会立即更新用户的访问权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 在 Users（用户）列表中，选择要删除其权限边界的 IAM 用户的名称。
5. 选择权限选项卡。如有必要，打开 Permissions boundary（权限边界）部分。

6. 选择 Change boundary（更改边界）。要确认要删除权限边界，请在确认对话框中选择 Remove boundary（删除边界）。

控制台会显示一条状态消息，通知您权限边界已删除。

添加和删除用户的权限（Amazon CLI 或 Amazon API）

要以编程方式添加或删除权限，您必须添加组成员资格、附加或分离托管策略或者添加或删除内联策略。有关更多信息，请参阅以下主题：

- 编辑 IAM 组中的用户
- 添加和删除 IAM 身份权限

Amazon 中的用户密码

您可以在账户中管理 IAM 用户的密码。IAM 用户需要密码才能访问 Amazon Web Services 管理控制台。利用 Amazon CLI、Tools for Windows PowerShell、Amazon SDK 或 API，用户无需密码即可以编程方式访问 Amazon 资源。在这些环境下，您可以选择为 IAM 用户分配访问密钥。但是，我们建议您首先考虑使用其他更安全的访问密钥替代方法。有关更多信息，请参阅 Amazon 安全凭证。

> **Note**
>
> 如果您的某个 IAM 用户丢失或忘记了密码，您无法从 IAM 中找回密码。根据您的设置，用户或管理员都必须创建新密码。

内容

- 为 IAM 用户设置账户密码策略
- 管理 IAM 用户的密码
- 允许 IAM 用户更改自己的密码
- IAM 用户如何更改自己的密码

为 IAM 用户设置账户密码策略

您可以在 Amazon Web Services 账户 上设置自定义密码策略，以便指定您的 IAM 用户密码的复杂性要求和强制轮换期。如果未设置自定义密码策略，则 IAM 用户密码必须符合默认 Amazon 密码策略。有关更多信息，请参阅 自定义密码策略选项。

主题

- 设置密码策略的规则
- 设置密码策略时所需的权限
- 默认密码策略
- 自定义密码策略选项
- 要设置密码策略（控制台）
- 更改密码策略（控制台）
- 删除自定义密码策略（控制台）
- 设置密码策略 (Amazon CLI)
- 设置密码策略 (Amazon API)

设置密码策略的规则

IAM 密码策略不适用于 Amazon Web Services 账户根用户 密码或 IAM 用户访问密钥。如果密码过期，IAM 用户将无法登录 Amazon Web Services 管理控制台，但可以继续使用其访问密钥。

创建或更改密码策略时，大多数密码策略设置会在用户下次更改其密码时实施。但是，一些设置将立即实施。例如：

- 在最小长度和字符类型要求变更后，系统会在您的用户下次更改其密码时强制实施该设置。不强制用户更改其现有密码，即使这些密码不符合更新后的密码策略。
- 设置密码有效期时，有效期立即生效。例如，假定您将密码有效期设置为 90 天。在这种情况下，对于现有密码使用期限超过 90 天的所有 IAM 用户，其密码将过期。这些用户必须在下次登录时更改其密码。

在尝试指定的登录失败次数后，您将无法创建“锁定策略”来锁定账户用户。为了增强安全性，我们建议您将强密码策略与 Multi-Factor Authentication (MFA) 相结合。有关 MFA 的更多信息，请参阅 IAM 中的 Amazon 多重身份验证。

设置密码策略时所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）查看或编辑其账户密码策略。您可以在 IAM policy 中包含以下密码策略操作：

- `iam:GetAccountPasswordPolicy` - 允许实体查看其账户的密码策略
- `iam:DeleteAccountPasswordPolicy` - 允许实体删除其账户的自定义密码策略并恢复到默认密码策略
- `iam:UpdateAccountPasswordPolicy` - 允许实体为其账户创建或更改自定义密码策略

以下策略允许查看和编辑账户密码策略的完全访问权限。要了解如何使用该示例 JSON 策略文档创建 IAM policy，请参阅。the section called “使用 JSON 编辑器创建策略”

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessPasswordPolicy",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:DeleteAccountPasswordPolicy",
                "iam:UpdateAccountPasswordPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 IAM 用户更改自己的密码所需权限的信息，请参阅 允许 IAM 用户更改自己的密码。

默认密码策略

如果管理员未设置自定义密码策略，则 IAM 用户密码必须符合默认 Amazon 密码策略。

默认密码策略强制执行以下条件：

- 密码长度最短为 8 个字符，最长为 128 个字符

- 至少包含以下三种字符类型的组合：大写、小写、数字，以及非字母数字字符（`! @ # $ % ^ & * ( ) _ + - = [ ] { } | '`）
- 与您的 Amazon Web Services 账户 名称或电子邮件地址不同
- 密码永不过期

自定义密码策略选项

为账户配置自定义密码策略时，可以指定以下条件：

- 密码最小长度 - 您可以指定最少 6 个字符和最多 128 个字符。
- 密码强度 - 您可以选中以下任一复选框来定义 IAM 用户密码的强度：
  - 至少需要一个大写拉丁字母（A-Z）
  - 至少需要一个小写拉丁字母（a-z）
  - 至少需要一个数字
  - 至少需要一个非字母数字字符 `! @ # $ % ^ & * ( ) _ + - = [ ] { } | '`
- Turn on password expiration（打开密码过期）– 您可以选择并指定 IAM 用户密码设置后的有效期：至少 1 天和最多 1095 天。例如，如果您指定的过期时间为 90 天，则会立即影响所有用户。对于密码超过 90 天的用户，更改后登录控制台时，必须设置新密码。密码为 75-89 天的用户会收到有关密码过期的 Amazon Web Services 管理控制台 警告。IAM 用户可以随时更改其密码（如有权限）。密码的有效期从用户设置新密码时起计算。IAM 用户同时只能有一个有效密码。
- Password expiration requires administrator reset（密码过期需要管理员重置）– 选择此选项可防止 IAM 用户在密码过期后使用 Amazon Web Services 管理控制台 更新自己的密码。选择此选项之前，请确认您的 Amazon Web Services 账户 具有多个具备管理权限的用户，以重置 IAM 用户密码。具有 `iam:UpdateLoginProfile` 权限的管理员可重置 IAM 用户密码。具有 `iam:ChangePassword` 权限和活动访问密钥的 IAM 用户可以编程方式重置其 IAM 用户控制台密码。如果清除此复选框，则密码已过期的 IAM 用户仍必须先设置新密码，然后才能访问 Amazon Web Services 管理控制台。
- Allow users to change their own password（允许用户更改自己的密码）– 您可以允许账户中的所有 IAM 用户更改自己的密码。这样用户就可以仅访问其用户的 `iam:ChangePassword` 操作和 `iam:GetAccountPasswordPolicy` 操作。此选项不会将权限策略附加到每个用户。相反，IAM 为所有用户提供账户级别权限。或者，您可以只允许部分用户管理自己的密码。为此，请清除此复选框。有关使用策略来限制哪些人可以管理密码的更多信息，请参阅允许 IAM 用户更改自己的密码。
- Prevent password reuse（防止密码重复使用）- 您可以阻止 IAM 用户重复使用指定数量的前密码。您可以指定最少 1 个和最多 24 个不能重复的前密码。

要设置密码策略（控制台）

您可使用 Amazon Web Services 管理控制台 创建、更改或删除自定义密码策略。密码策略的更改适用于在此策略更改后创建的新 IAM 用户，以及更改密码时的现有 IAM 用户。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Account settings（账户设置）。
4. 在 Password policy（密码策略）部分，选择 Edit（编辑）。
5. 选择 Custom（自定义）以使用自定义密码策略。
6. 选择您要应用于密码策略的选项，然后选择 Save changes（保存更改）。
7. 通过选择 Set custom（设置自定义）来确认您要设置自定义密码策略。

控制台会显示一条状态消息，通知您 IAM 用户的密码要求已更新。

更改密码策略（控制台）

您可使用 Amazon Web Services 管理控制台 创建、更改或删除自定义密码策略。密码策略的更改适用于在此策略更改后创建的新 IAM 用户，以及更改密码时的现有 IAM 用户。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Account settings（账户设置）。
4. 在 Password policy（密码策略）部分，选择 Edit（编辑）。
5. 选择您要应用于密码策略的选项，然后选择 Save changes（保存更改）。
6. 通过选择 Set custom（设置自定义）来确认您要设置自定义密码策略。

控制台会显示一条状态消息，通知您 IAM 用户的密码要求已更新。

删除自定义密码策略（控制台）

您可使用 Amazon Web Services 管理控制台 创建、更改或删除自定义密码策略。密码策略的更改适用于在此策略更改后创建的新 IAM 用户，以及更改密码时的现有 IAM 用户。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Account settings（账户设置）。
4. 在 Password policy（密码策略）部分，选择 Edit（编辑）。
5. 选择 IAM default（IAM 默认）以删除自定义密码策略，然后选择 Save changes（保存更改）。
6. 通过选择 Set default（设置默认值）来确认您要设置 IAM 默认密码策略。

   控制台会显示一条状态消息，通知您密码策略已设置为 IAM 默认值。

设置密码策略 (Amazon CLI)

您可以使用 Amazon Command Line Interface 设置密码策略。

通过 Amazon CLI 管理自定义账户密码策略

运行以下 命令：

- 创建或更改自定义密码策略：`aws iam update-account-password-policy`
- 查看密码策略：`aws iam get-account-password-policy`
- 删除自定义密码策略：`aws iam delete-account-password-policy`

设置密码策略 (Amazon API)

您可以使用 Amazon API 操作来设置密码策略。

通过 Amazon API 管理自定义账户密码策略

调用以下操作：

- 创建或更改自定义密码策略：`UpdateAccountPasswordPolicy`

- 查看密码策略：GetAccountPasswordPolicy
- 删除自定义密码策略：DeleteAccountPasswordPolicy

管理 IAM 用户的密码

通过 Amazon Web Services 管理控制台 使用 Amazon 资源的 IAM 用户必须具有密码才能登录。您可以创建、更改或删除您的 Amazon 账户中 IAM 用户的密码。

向用户分配密码后，该用户可使用您的账户的登录 URL 登录 Amazon Web Services 管理控制台，如下所示：

```
https://12-digit-Amazon-account-ID or alias.signin.aws.amazon.com/console
```

有关 IAM 用户如何登录 Amazon Web Services 管理控制台 的更多信息，请参阅《Amazon 登录 用户指南》中的如何登录 Amazon。

即使用户有自己的密码，还是需要权限才能访问您的 Amazon 资源。默认情况下，用户没有权限。为授予用户所需的许可，您可向用户或用户所属的群组分配策略。有关创建用户和组的信息，请参阅 IAM 身份。有关使用策略来设置许可的信息，请参阅 更改 IAM 用户的权限。

可向用户授予相应的权限，用于更改其自身的密码。有关更多信息，请参阅 允许 IAM 用户更改自己的密码。有关用户如何访问账户登录页面的信息，请参阅《Amazon 登录 用户指南》中的如何登录 Amazon。

主题

- 创建、更改或删除 IAM 用户密码（控制台）

创建、更改或删除 IAM 用户密码（控制台）

您可以使用 Amazon Web Services 管理控制台 管理 IAM 用户的密码。

用户的访问需求可能会随着时间而变化。您可能需要让原本拥有 CLI 访问权限的用户开始访问控制台，因为用户收到了包含其凭证的电子邮件而更改其密码，或者在用户离开您的组织或不再需要 Amazon 访问权限时删除其密码。

要创建 IAM 用户密码（控制台）

使用此步骤创建与用户名相关联的密码，为用户提供控制台访问权限。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 请选择要为其创建密码的用户的名称。
5. 选择 Security credentials（安全凭证）选项卡，然后在 Console sign-in（控制台登录）下选择 Enable console access（启用控制台访问权限）。
6. 在 Enable console access（启用控制台访问权限）对话框中，选择 Reset password（重置密码），然后选择是让 IAM 生成密码还是创建自定义密码：
   - 要让 IAM 生成密码，请选择 Autogenerated password（自动生成的密码）。
   - 要创建自定义密码，请选择 Custom password（自定义密码），然后键入密码。

   > **Note**
   > 您创建的密码必须符合账户的密码策略。
7. 若要求用户在登录时创建新密码，请选择 Require password change at the next sign-in（必须在下次登录时更改密码）。
8. 若要求用户立即使用新密码，请选择 Revoke active console sessions（撤消活动控制台会话）。这会向 IAM 用户附加一个内联策略，如果用户的凭证超过了策略指定的时间，则该策略就会拒绝用户访问资源。
9. 选择 Reset password（重置密码）。
10. Console password（控制台密码）对话框通知您已启用用户的新密码。要查看密码以便与用户共享它，请在 Console password（控制台密码）对话框中选择 Show（显示）。选择 Download .csv file（下载 .csv 文件），下载包含用户凭证的文件。

    > **Important**
    > 出于安全原因，在完成该步骤后您无法访问该密码，但您可以随时创建新密码。

控制台会显示一条状态消息，通知您控制台访问权限已启用。

要更改 IAM 用户的密码（控制台）

使用此步骤更新与用户名关联的密码。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 请选择要更改其密码的用户的名称。
5. 选择 Security credentials（安全凭证）选项卡，然后在 Console sign-in（控制台登录）下选择 Manage console access（管理控制台访问权限）。
6. 在 Manage console access（管理控制台访问权限）对话框中，选择 Reset password（重置密码），然后选择是让 IAM 生成密码还是创建自定义密码：
   - 要让 IAM 生成密码，请选择 Autogenerated password（自动生成的密码）。
   - 要创建自定义密码，请选择 Custom password（自定义密码），然后键入密码。

   > **Note**
   > 您创建的密码必须符合账户的密码策略。
7. 若要求用户在登录时创建新密码，请选择 Require password change at the next sign-in（必须在下次登录时更改密码）。
8. 若要求用户立即使用新密码，请选择 Revoke active console sessions（撤消活动控制台会话）。这会向 IAM 用户附加一个内联策略，如果用户的凭证超过了策略指定的时间，则该策略就会拒绝用户访问资源。
9. 选择 Reset password（重置密码）。
10. Console password（控制台密码）对话框通知您已启用用户的新密码。要查看密码以便与用户共享它，请在 Console password（控制台密码）对话框中选择 Show（显示）。选择 Download .csv file（下载 .csv 文件），下载包含用户凭证的文件。

    > **Important**
    > 出于安全原因，在完成该步骤后您无法访问该密码，但您可以随时创建新密码。

控制台会显示一条状态消息，通知您控制台访问权限已更新。

要删除（禁用）IAM 用户的密码（控制台）

使用此步骤删除与用户名关联的密码，从而取消用户的控制台访问权限。

> **Important**
>
> 您可以通过删除 IAM 用户的密码来阻止其访问 Amazon Web Services 管理控制台。这样可以阻止他们使用其登录凭证登录 Amazon Web Services 管理控制台。这不会更改他们的权限，也不会阻止他们使用担任的角色访问控制台。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell、Amazon API，或 Amazon Console Mobile Application 进行访问。

Console

1. 按照 Amazon 登录用户指南中的如何登录 Amazon所述，根据用户类型选择相应的登录过程。
2. 在 IAM 控制台主页左侧的导航窗格中，在搜索 IAM 文本框中输入查询语句。
3. 在导航窗格中，选择 Users（用户）。
4. 请选择要删除其密码的用户的名称。
5. 选择安全凭证选项卡，然后在控制台登录下选择管理控制台访问权限。
6. 若要求用户立即停止使用控制台，请选择 Revoke active console sessions（撤消活动控制台会话）。这会向 IAM 用户附加一个内联策略，如果用户的凭证超过了策略指定的时间，则该策略就会拒绝用户访问资源。
7. 然后选择 Disable access（禁用访问）。

控制台会显示一条状态消息，通知您控制台访问已禁用。

创建、更改或删除 IAM 用户密码 (Amazon CLI)

您可以使用 Amazon CLI 管理 IAM 用户的密码。

创建密码 (Amazon CLI)

1. （可选）要确定用户是否有密码，请运行此命令：aws iam get-login-profile

2. 要创建密码，请运行此命令：aws iam create-login-profile

更改用户的密码 (Amazon CLI)

1. （可选）要确定用户是否有密码，请运行此命令：aws iam get-login-profile
2. 要更改密码，请运行此命令：aws iam update-login-profile

删除（禁用）用户的密码 (Amazon CLI)

1. （可选）要确定用户是否有密码，请运行此命令：aws iam get-login-profile
2. （可选）要确定上次使用密码的时间，请运行此命令：aws iam get-user
3. 要删除密码，请运行此命令：aws iam delete-login-profile

> **Important**
>
> 在删除用户的密码后，用户无法再登录到 Amazon Web Services 管理控制台。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 函数调用进行访问。在使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 从您的 Amazon Web Services 账户 中删除用户时，您必须先使用此操作删除密码。有关更多信息，请参阅 移除 IAM 用户 (Amazon CLI)。

在指定时间之前撤销用户的活动控制台会话 (Amazon CLI)

1. 要嵌入在指定时间之前撤销 IAM 用户的活动控制台会话的内联策略，请使用以下内联策略并运行此命令：aws iam put-user-policy

   此内联策略拒绝所有权限并包含 `aws:TokenIssueTime` 条件键。它会在内联策略的 `Condition` 元素中的指定时间之前撤销用户的活动控制台会话。将 `aws:TokenIssueTime` 条件键值替换为您自己的值。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
```

```
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {
        "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
      }
    }
  }
}
```

2. （可选）要列出 IAM 用户中嵌入的内联策略的名称，请运行此命令：aws iam list-user-policies
3. （可选）要查看 IAM 用户中嵌入的命名内联策略，请运行此命令：aws iam get-user-policy

创建、更改或删除 IAM 用户密码 (Amazon API)

您可以使用 Amazon API 管理 IAM 用户的密码。

创建密码 (Amazon API)

1. （可选）要确定用户是否有密码，请调用此操作：GetLoginProfile
2. 要创建密码，请调用此操作：CreateLoginProfile

更改用户的密码 (Amazon API)

1. （可选）要确定用户是否有密码，请调用此操作：GetLoginProfile
2. 要更改密码，请调用此操作：UpdateLoginProfile

删除（禁用）用户的密码 (Amazon API)

1. （可选）要确定用户是否有密码，请运行此命令：GetLoginProfile
2. （可选）要确定上次使用密码的时间，请运行此命令：GetUser
3. 要删除密码，请运行此命令：DeleteLoginProfile

> **Important**
>
> 在删除用户的密码后，用户无法再登录到 Amazon Web Services 管理控制台。如果该用户具有有效的访问密钥，它们将继续起作用并允许通过 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 函数调用进行访问。在使用 Amazon CLI、Tools for Windows

PowerShell 或 Amazon API 从您的 Amazon Web Services 账户 中删除用户时，您必须先使用此操作删除密码。有关更多信息，请参阅 移除 IAM 用户 (Amazon CLI)。

在指定时间之前撤销用户的活动控制台会话 (Amazon API)

1. 要嵌入在指定时间之前撤销 IAM 用户的活动控制台会话的内联策略，请使用以下内联策略并运行此命令：PutUserPolicy

   此内联策略拒绝所有权限并包含 `aws:TokenIssueTime` 条件键。它会在内联策略的 `Condition` 元素中的指定时间之前撤销用户的活动控制台会话。将 `aws:TokenIssueTime` 条件键值替换为您自己的值。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {
        "aws:TokenIssueTime": "2014-05-07T23:47:00Z"
      }
    }
  }
}
```

2. （可选）要列出 IAM 用户中嵌入的内联策略的名称，请运行此命令：ListUserPolicies
3. （可选）要查看 IAM 用户中嵌入的命名内联策略，请运行此命令：GetUserPolicy

允许 IAM 用户更改自己的密码

> **Note**
>
> 具有联合身份的用户将使用其身份提供商定义的流程来更改密码。作为最佳实践，要求人类用户使用带有身份提供商的联合身份验证才能使用临时凭证访问 Amazon。

您可以向 IAM 用户授予更改其用于登录 Amazon Web Services 管理控制台 的密码的权限。您可以通过两种方式之一来执行此操作：

- 允许账户中的所有 IAM 用户更改自己的密码。
- 仅允许选定的 IAM 用户更改自己的密码。在此情况下，您禁用可供所有用户用来更改其密码的选项，并使用 IAM policy 仅向某些用户授予权限。这种方法允许这些用户更改自己的密码和可选的其他凭证，例如他们自己的访问密钥。

> **Important**
>
> 我们建议您设置自定义密码策略，要求 IAM 用户创建强密码。

允许所有 IAM 用户更改自己的密码

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，单击 Account settings（账户设置）。
3. 在 Password policy（密码策略）部分，选择 Edit（编辑）。
4. 选择 Custom（自定义）以使用自定义密码策略。
5. 选择 Allow users to change their own password（允许用户更改其密码），然后选择 Save changes（保存更改）。这样账户中的所有用户就可以仅访问其用户的 `iam:ChangePassword` 操作和 `iam:GetAccountPasswordPolicy` 操作。
6. 向用户提供有关更改密码的以下说明：IAM 用户如何更改自己的密码。

Amazon CLI

运行以下命令：

- `aws iam update-account-password-policy`

API

要更新账户密码策略，请调用以下操作：

- UpdateAccountPasswordPolicy

允许选定的 IAM 用户更改自己的密码

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，单击 Account settings（账户设置）。
3. 在 Password policy（密码策略）部分中，确保未选择 Allow users to change their own password（允许用户更改其密码）。如果已选中此复选框，则所有用户均可更改其密码。（请参阅上一个过程。）
4. 创建允许更改其密码的用户（如果之前没有这样的用户）。有关更多信息，请参阅 在 Amazon Web Services 账户 中创建 IAM 用户。
5. （可选）为可以更改自己密码的用户创建 IAM 组，然后将上一步骤中的用户添加到该组。有关更多信息，请参阅 IAM 用户组。
6. 将下列策略分配到该组。有关更多信息，请参阅 管理 IAM 策略。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:GetAccountPasswordPolicy",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:ChangePassword",
      "Resource": "arn:aws:iam::*:user/${aws:username}"
    }
  ]
}
```

此策略授予对 ChangePassword 操作的访问权限，这让用户可从控制台、Amazon CLI、Tools for Windows PowerShell 或 API 仅更改他们自己的密码。它还授予对 GetAccountPasswordPolicy 操作的访问权限，这可让用户查看当前的密码策略。用户需要此权限才能在 Change password（更改密码）页面上查看账户密码策略。用户必须能阅读当前密码策略以确保更改后的密码符合策略的要求。

7. 向用户提供有关更改密码的以下说明：IAM 用户如何更改自己的密码。

有关更多信息

有关管理凭证的更多信息，请参阅以下主题：

- 允许 IAM 用户更改自己的密码
- Amazon 中的用户密码
- 为 IAM 用户设置账户密码策略
- 管理 IAM 策略
- IAM 用户如何更改自己的密码

IAM 用户如何更改自己的密码

如果您已获得更改自己 IAM 用户密码的权限，则可以使用 Amazon Web Services 管理控制台 中的特定页面完成此操作。您还可以使用 Amazon CLI 或 Amazon API。

主题

- 所需权限
- IAM 用户如何更改自己的密码（控制台）
- IAM 用户如何更改自己的密码（Amazon CLI 或 Amazon API）

所需权限

要为您自己的 IAM 用户更改密码，您必须具有以下策略中的权限：Amazon：允许 IAM 用户在的“安全凭证”页面上更改自己的控制台密码。

IAM 用户如何更改自己的密码（控制台）

以下过程介绍了 IAM 用户如何使用 Amazon Web Services 管理控制台 更改自己的密码。

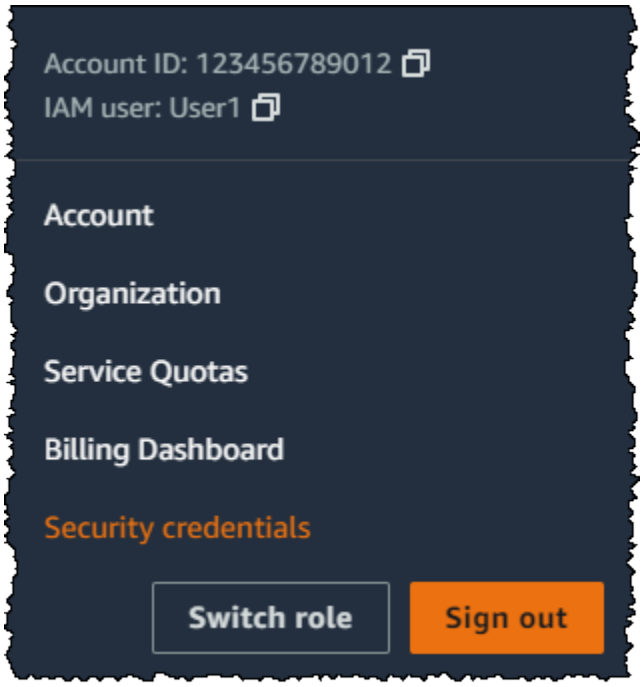
更改您自己的 IAM 用户密码（控制台）

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

   > **Note**
   >
   > 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

   要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Amazon IAM 凭证选项卡上，选择更改密码。
4. 对于 Current password (当前密码)，请键入您的当前密码。在 New password (新密码) 和 Confirm new password (确认新密码) 框中键入新密码。然后选择更新密码。

> **Note**
>
> 新密码必须符合账户密码策略的要求。有关更多信息，请参阅 为 IAM 用户设置账户密码策略。

IAM 用户如何更改自己的密码（Amazon CLI 或 Amazon API）

以下过程介绍了 IAM 用户如何使用 Amazon CLI 或 Amazon API 更改自己的密码。

要更改自己的 IAM 密码，请使用以下命令：

- Amazon CLI: `aws iam change-password`
- Amazon API：`ChangePassword`

管理 IAM 用户的访问密钥。

> **Important**
>
> 作为一项 最佳实践，请使用临时安全凭证（例如 IAM 角色）而非创建访问密钥等长期凭证。在创建访问密钥之前，请认真阅读 长期访问密钥的替代方法。

访问密钥是 IAM 用户或 Amazon Web Services 账户根用户 的长期凭证。您可以使用访问密钥签署对 Amazon CLI 或 Amazon API 的编程请求（直接或使用 Amazon 开发工具包）。有关更多信息，请参阅 使用 Amazon 安全凭证以编程方式进行访问。

访问密钥包含两部分：访问密钥 ID（例如 `AKIAIOSFODNN7EXAMPLE`）和秘密访问密钥（例如 `wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY`）。您必须同时使用访问密钥 ID 和秘密访问密钥对请求执行身份验证。

当您创建访问密钥对时，将访问密钥 ID 和秘密访问密钥保存在一个安全位置。秘密访问密钥只能在创建密钥时检索到。如果您丢失了秘密访问密钥，则必须删除访问密钥并创建新的访问密钥。有关更多说明，请参阅 更新访问密钥。

每个用户最多可有两个访问密钥。

> **Important**
>
> 拥有访问密钥的 IAM 用户存在账户安全风险。安全管理您的访问密钥。请不要向未经授权方提供访问密钥，即便是为了帮助找到您的账户标识符也不行。如果您这样做，可能会向某人提供对您的账户的永久访问权限。
> 下面是使用访问密钥时需要注意的概念：
>
> - 请勿使用账户的根凭证来创建访问密钥。
> - 请勿在应用程序文件中放置访问密钥或凭证信息。
> - 请勿在项目区域中放置包含访问密钥或凭证信息的文件。
> - 在共享 Amazon 凭证中存储的访问密钥或凭证信息以明文形式存储。

监控建议

创建访问密钥后：

- 使用 Amazon CloudTrail 监控访问密钥的使用情况，并检测任何未经授权的访问尝试。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。
- 设置 CloudWatch 警报，在访问尝试被拒绝时通知管理员，帮助检测恶意活动。有关更多信息，请参阅《Amazon CloudWatch 用户指南》。
- 根据需要定期检查、更新和删除访问密钥。

以下主题详细介绍了与访问密钥相关的管理任务。

主题

- 通过将内联策略附加到 IAM 用户，从而控制访问密钥的使用
- 管理访问密钥所需的权限
- IAM 用户如何管理自己的访问密钥
- IAM 管理员如何管理 IAM 用户访问密钥
- 更新访问密钥
- 保护访问密钥

通过将内联策略附加到 IAM 用户，从而控制访问密钥的使用

建议最好让工作负载使用带有 IAM 角色的临时凭证来访问 Amazon。应为拥有访问密钥的 IAM 用户分配最低访问权限，并启用多重身份验证 (MFA)。有关代入 IAM 角色的更多信息，请参阅担任角色的方法。

但是，如果您正在为服务自动化或其他短期使用案例创建概念验证测试，并选择使用具有访问密钥的 IAM 用户来运行工作负载，建议使用策略条件来进一步限制对其 IAM 用户凭证的访问权限。

在此情况下，您可以创建有时限的策略，使凭证在指定时间后过期，或者如果您是从安全网络运行工作负载，可以使用 IP 限制策略。

对于这两个用例，可使用附加到拥有访问密钥的 IAM 用户的内联策略。

为 IAM 用户配置有时限的策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择用户，然后为短期用例选择用户。如果尚未创建用户，您可以立即创建用户。
3. 在角色的详细信息页面上，选择权限选项卡。
4. 选择添加权限，然后选择创建内联策略。
5. 在策略编辑器部分，选择 JSON 显示 JSON 编辑器。
6. 在 JSON 编辑器中输入以下策略，将 `aws:CurrentTime` 时间戳的值替换为所需的到期日期和时间：

   JSON

```
{
"Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
      "DateGreaterThan": {
      "aws:CurrentTime": "2025-03-01T00:12:00Z"
        }
      }
```

```
    }
  ]
}
```

该策略使用 Deny 效果来限制在指定日期后对所有资源执行所有操作。DateGreaterThan 条件比较当前的时间与您设置的时间戳。

7. 选择下一步，进入查看并创建页面。在策略详细信息的策略名称下，输入策略的名称，然后选择创建策略。

创建策略后，策略将显示在用户的权限选项卡上。当前时间大于或等于该策略中指定的时间时，用户就无法再访问 Amazon 资源。请务必让工作负载开发人员了解您为这些访问密钥指定的到期日期。

为 IAM 用户配置 IP 限制策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择用户，然后再选择将从安全网络运行工作负载的用户。如果尚未创建用户，您可以立即创建用户。
3. 在角色的详细信息页面上，选择权限选项卡。
4. 选择添加权限，然后选择创建内联策略。
5. 在策略编辑器部分，选择 JSON 显示 JSON 编辑器。
6. 将以下 IAM 策略复制到 JSON 编辑器，并根据需要更改公有 IPv4 或 IPv6 地址或范围。可使用 https://checkip.amazonaws.com 确定当前的公有 IP 地址。您可以使用斜杠标记指定单个 IP 地址或 IP 地址范围。有关更多信息，请参阅 aws:SourceIp。

> **Note**
>
> 不得通过 VPN 或代理服务器对 IP 地址进行模糊处理。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Sid":"IpRestrictionIAMPolicyForIAMUser",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "203.0.113.0/24",
            "2001:DB8:1234:5678::/64",
            "203.0.114.1"
          ]
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

此策略示例拒绝使用应用该策略的 IAM 用户访问密钥，除非请求来自（以 CIDR 表示法指定的）网络“203.0.113.0/24”、“2001:DB8:1234:5678::/64”或特定 IP 地址“203.0.114.1”

7. 选择下一步，进入查看并创建页面。在策略详细信息的策略名称下，输入策略的名称，然后选择创建策略。

创建策略后，策略将显示在用户的权限选项卡上。

您还可以将该策略作为服务控制策略 (SCP) 应用于 Amazon Organizations 中的多个 Amazon 账户，我们建议使用附加条件 `aws:PrincipalArn` 让该策略声明仅适用于受此 SCP 约束的 Amazon 账户中的 IAM 用户。以下策略包括该更新：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IpRestrictionServiceControlPolicyForIAMUsers",
      "Effect": "Deny",
```

```
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "NotIpAddress": {
          "aws:SourceIp": [
            "203.0.113.0/24",
            "2001:DB8:1234:5678::/64",
            "203.0.114.1"
          ]
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        },
        "ArnLike": {
          "aws:PrincipalArn": "arn:aws:iam::*:user/*"
        }
      }
    }
  ]
}
```

管理访问密钥所需的权限

> **Note**
>
> `iam:TagUser` 是用于添加和编辑访问密钥描述的可选权限。有关更多信息，请参阅 标记 IAM 用户。

要为您自己的 IAM 用户创建访问密钥，您必须具有以下策略中的权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
```

```
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

要为您自己的 IAM 用户更新访问密钥，您必须具有以下策略中的权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:GetAccessKeyLastUsed",
                "iam:GetUser",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:TagUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

IAM 用户如何管理自己的访问密钥

IAM 管理员可以通过附加 管理访问密钥所需的权限 中所属的策略，授予 IAM 用户自我管理访问密钥的权限。

有了这些权限，IAM 用户就可以通过以下步骤来创建、激活、停用和删除与其用户名相关联的访问密钥。

主题

- 为自己创建访问密钥（控制台）
- 停用您的访问密钥（控制台）
- 激活您的访问密钥（控制台）
- 删除您的访问密钥（控制台）

为自己创建访问密钥（控制台）

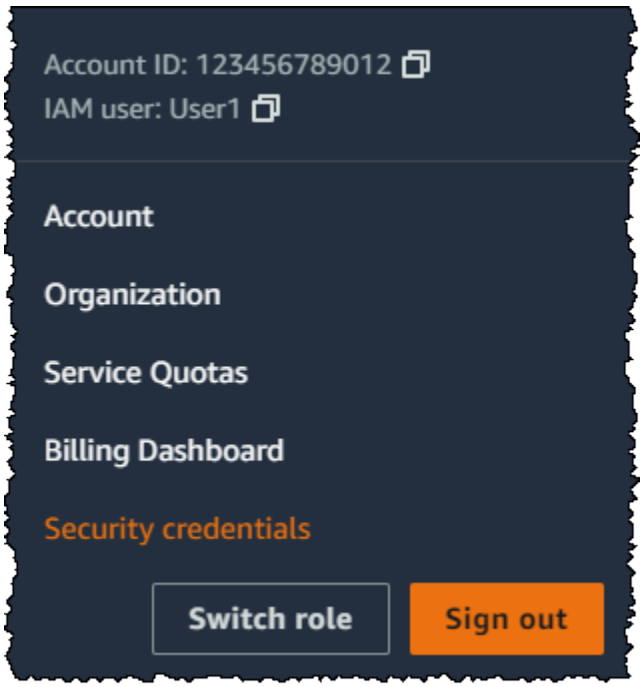
如果您已获得相应的权限，则可以使用 Amazon Web Services 管理控制台 为自己创建访问密钥。

要为自己创建访问密钥（控制台）

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

   > **Note**
   >
   > 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

   要获取 Amazon Web Services 账户 ID，请联系管理员。
2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在访问密钥部分，选择创建访问密钥。如果您已经有两个访问密钥，则此按钮将被停用，您必须先删除一个访问密钥，然后才能创建新的访问密钥。
4. 在 Access key best practices & alternatives（访问密钥最佳实践和替代方法）页面上，请选择您的用例以了解可帮助您避免创建长期访问密钥的其他选项。如果您确定您的用例仍然需要访问密钥，请选择 Other（其他），然后选择 Next（下一步）。
5. （可选）为访问密钥设置描述标记值。这会为您的 IAM 用户添加标签键/值对。这有助于您以后标识和更新访问密钥。标签密钥设置为访问密钥 ID。标签值设置为您指定的访问密钥描述。完成后，选择 Create access key（创建访问密钥）。
6. 在 Retrieve access keys（检索访问密钥）页面上，选择 Show（显示）来显示用户的秘密访问密钥的值，或选择 Download .csv file（下载 .csv 文件）。这是您保存秘密访问密钥的唯一机会。将秘密访问密钥保存在安全位置后，请选择 Done（完成）。

停用您的访问密钥（控制台）

如果您已获得相应的权限，则可以使用 Amazon Web Services 管理控制台 来停用您的访问密钥。

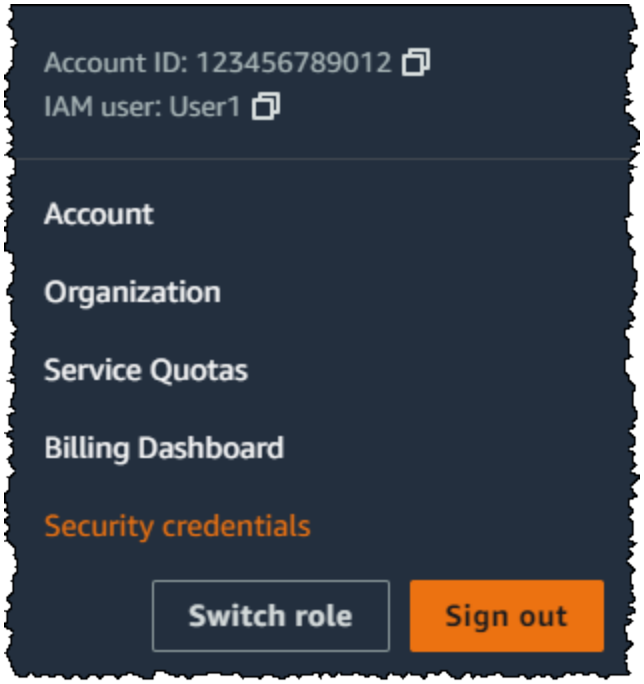
停用访问密钥

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

> **Note**
>
> 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Access keys（访问密钥）部分中，找到要停用的密钥，选择 Actions（操作），然后选择 Deactivate（停用）。当系统提示您确认时，请选择 Deactivate (停用）。已停用的访问密钥仍会计入您的两个访问密钥限制。

激活您的访问密钥（控制台）

如果您已获得相应的权限，则可以使用 Amazon Web Services 管理控制台 来激活您的访问密钥。

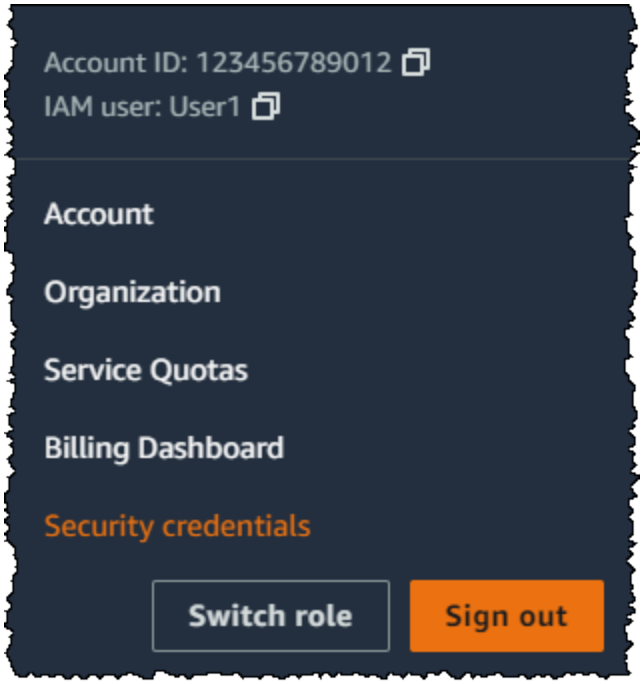
激活访问密钥

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

> **Note**
>
> 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Access keys（访问密钥）部分中，找到要激活的密钥，选择 Actions（操作），然后选择 Activate（激活）。

删除您的访问密钥（控制台）

如果您已获得相应的权限，则可以使用 Amazon Web Services 管理控制台 来删除您的访问密钥。

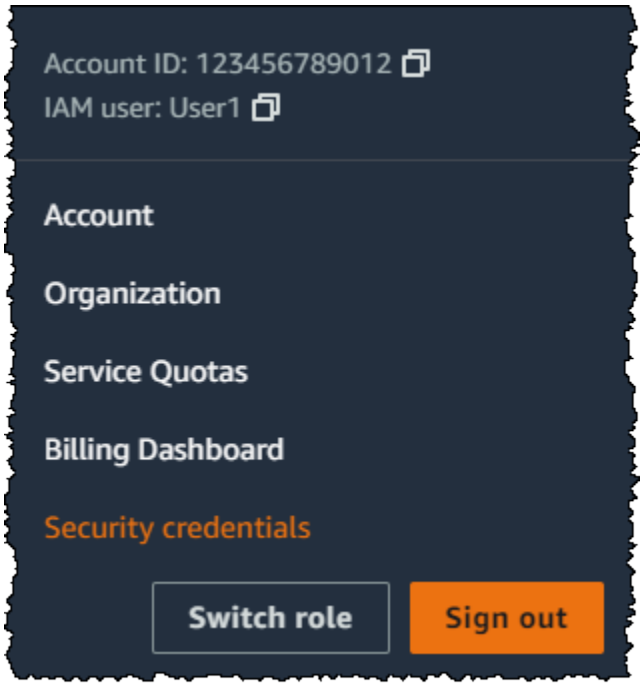
删除不再需要的访问密钥

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

> **Note**
>
> 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Access keys（访问密钥）部分中，找到要删除的密钥，选择 Actions（操作），然后选择 Delete（删除）。按照对话框中的说明先 Deactivate（停用），然后确认删除。我们建议您在永久删除访问密钥之前验证该访问密钥是否已不再使用。

IAM 管理员如何管理 IAM 用户访问密钥

IAM 管理员可以创建、激活、停用和删除与单个 IAM 用户关联的访问密钥。他们还可以列出账户中拥有访问密钥的 IAM 用户，并找出拥有特定访问密钥的 IAM 用户。

主题

- 为 IAM 用户创建访问密钥

- 要停用 IAM 用户的访问密钥
- 要激活 IAM 用户的访问密钥
- 删除 IAM 用户的访问密钥
- 要列出 IAM 用户的访问密钥
- 要显示您账户中用户的所有访问密钥 ID
- 要使用访问密钥 ID 查找用户
- 要查找访问密钥 ID 的最新使用情况

为 IAM 用户创建访问密钥

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 在 Security credentials（安全凭证）选项卡的 Access keys（访问密钥）部分中，选择 Create access key（创建访问密钥）。

   如果按钮已停用，则必须先删除现有密钥中的一个，然后才能创建新密钥。
5. 在 Access key best practices & alternatives（访问密钥最佳实践和替代方法）页面上，查看最佳实践和替代方法。选择您的用例以了解可帮助您避免创建长期访问密钥的其他选项。
6. 如果您确定您的用例仍然需要访问密钥，请选择 Other（其他），然后选择 Next（下一步）。
7. （可选）在设置描述标签页面上，您可以向访问密钥添加描述标签，以帮助跟踪您的访问密钥。选择创建访问密钥。
8. 在 Retrieve access keys（检索访问密钥）页面上，选择 Show（显示）来显示用户的秘密访问密钥的值。
9. 要将访问密钥 ID 和秘密访问密钥以 `.csv` 文件形式保存计算机上的安全位置，请选择 Download .csv file（下载 .csv 文件）按钮。

> **Important**
>
> 这是您查看或下载新创建的访问密钥的唯一机会，您将无法恢复访问密钥。确保安全地维护您的访问密钥。

为用户创建访问密钥时，默认情况下，密钥对处于活动状态，并且您可以立即使用此密钥对。

Amazon CLI

运行以下命令：

- aws iam create-access-key

API

调用以下操作：

- CreateAccessKey

要停用 IAM 用户的访问密钥

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 在 Security credentials（安全凭证）选项卡的 Access keys（访问密钥）部分，选择 Actions（操作）下拉菜单，然后选择 Deactivate（停用）。
5. 在 Deactivate（停用）对话框中，选择 Deactivate（停用），确认要停用访问密钥。

停用访问密钥后，API 调用将不再使用该密钥。如果需要，您可以再次激活它。

Amazon CLI

运行以下命令：

- aws iam update-access-key

API

调用以下操作：

- UpdateAccessKey

要激活 IAM 用户的访问密钥

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 在 Security credentials（安全凭证）选项卡的 Access keys（访问密钥）部分，选择 Actions（操作）下拉菜单，然后选择 Activate（激活）。

激活访问密钥后，API 调用可以使用该密钥。如果需要，您可以再次停用它。

Amazon CLI

运行以下命令：

- `aws iam update-access-key`

API

调用以下操作：

- `UpdateAccessKey`

删除 IAM 用户的访问密钥

停用访问密钥后，如果不再需要它，请将其删除。

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 在 Security credentials（安全凭证）选项卡的 Access keys（访问密钥）部分，选择非活动访问密钥的 Actions（操作）下拉菜单，然后选择 Delete（删除）。

5. 在 Delete（删除）对话框中，通过在文本输入字段中输入访问密钥 ID，然后选择 Delete（删除），确认要删除访问密钥。

访问密钥删除后，将无法恢复。

Amazon CLI

运行以下命令：

- `aws iam delete-access-key`

API

调用以下操作：

- `DeleteAccessKey`

要列出 IAM 用户的访问密钥

您可以查看与 IAM 用户关联的访问密钥 ID 的列表。

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 在安全凭证选项卡中的访问密钥部分列出了用户的访问密钥 ID，包括显示的每个密钥的状态。

> **Note**
>
> 只有用户的访问密钥 ID 是可见的。秘密访问密钥只能在创建密钥时检索到。

每个 IAM 用户最多可以拥有两个访问密钥。

Amazon CLI

运行以下命令：

- `aws iam list-access-keys`

API

调用以下操作：

- `ListAccessKeys`

要显示您账户中用户的所有访问密钥 ID

您可以查看 Amazon Web Services 账户 中用户的访问密钥 ID 列表。

Console

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择用户名称转到用户详细信息页面。
4. 如有必要，可通过完成以下步骤来将 Access key ID 列添加到用户表中：
   a. 在表格最右侧上方，选择 Preferences（首选项）图标 ( )。
   b. 在 Preferences（首选项）对话框的 Select visible columns（选择可见列）下，打开 Access key ID（访问密钥 ID）。
   c. 选择确认，返回到用户列表。更新列表以包含访问密钥 ID。
5. Access key ID（访问密钥 ID）列显示每个访问密钥的状态，后跟其 ID；例如，**`Active - AKIAIOSFODNN7EXAMPLE`** 或 **`Inactive - AKIAI44QH8DHBEXAMPLE`**。

   您可以利用该信息查看和复制具有一个或两个访问密钥的用户的访问密钥 ID。对于没有访问密钥的用户，该列显示 **`-`**。

   > **Note**
   > 秘密访问密钥只能在创建密钥时检索到。

每个 IAM 用户最多可以拥有两个访问密钥。

要使用访问密钥 ID 查找用户

您可以使用访问密钥 ID 来查找您 Amazon Web Services 账户 中的用户。

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格的搜索框中，输入访问密钥 ID，例如 AKIAI44QH8DHBEXAMPLE。
3. 与访问密钥 ID 关联的 IAM 用户将显示在导航窗格中。选择用户名称转到用户详细信息页面。

要查找访问密钥 ID 的最新使用情况

访问密钥的最新使用情况会显示在用户详情页面中 IAM 用户页面的用户列表中，并且是凭证报告的一部分。

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在用户列表中，参阅 Access key last used（上次使用的访问密钥）列。

   如果未显示该列，请选择 Preferences（首选项）图标
   ( )，
   然后在 Select visible columns（选择可见列）下打开 Access key last used（上次使用的访问密钥），以显示列。
3. （可选）在导航窗格的 Access reports（访问报告）下，选择 Credential report（凭证报告）以下载一份报告，其中包含您账户中所有 IAM 用户的访问密钥的上次使用信息。
4. (可选）选择 IAM 用户以查看用户详细信息。Summary（摘要）部分包括访问密钥 ID、其状态以及上次使用时间。

Amazon CLI

运行以下命令：

- `aws iam get-access-key-last-used`

API

调用以下操作：

- GetAccessKeyLastUsed

更新访问密钥

作为一项安全最佳实践，建议您在需要时更新 IAM 用户访问密钥，例如员工从公司离职时。已获得必要权限的 IAM 用户可以更新自己的访问密钥。

有关如何向 IAM 用户授予自行更新访问密钥的详细信息，请参阅 Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的密码、访问密钥和 SSH 公有密钥。您还可以将密码策略应用于您的账户，以要求所有 IAM 用户定期更新其密码，并规定必须执行这一操作的频率。有关更多信息，请参阅 为 IAM 用户设置账户密码策略。

> **Note**
>
> 如果您丢失了秘密访问密钥，则必须删除访问密钥并创建新的访问密钥。秘密访问密钥只能在创建密钥时检索到。使用此过程停用所有丢失的访问密钥，然后用新的凭证替换任何丢失的访问密钥。

主题

- 更新 IAM 用户访问密钥（控制台）
- 更新访问密钥（Amazon CLI）
- 更新访问密钥（Amazon API）

更新 IAM 用户访问密钥（控制台）

您可以通过 Amazon Web Services 管理控制台更新访问密钥。

在不中断应用程序的情况下更新 IAM 用户的访问密钥（控制台）

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥。
   a. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
   b. 在导航窗格中，选择用户。

c. 选择预期用户的名称，然后选择安全凭证选项卡。

d. 在访问密钥部分，选择创建访问密钥。在 Access key best practices & alternatives（访问密钥最佳实践和替代方法）页面上，选择 Other（其他），然后选择 Next（下一步）。

e. （可选）设置访问密钥的描述标签值，以向 IAM 用户添加标签键/值对。这有助于您以后标识和更新访问密钥。标签密钥设置为访问密钥 ID。标签值设置为您指定的访问密钥描述。完成后，选择 Create access key（创建访问密钥）。

f. 在 Retrieve access keys（检索访问密钥）页面上，选择 Show（显示）来显示用户的秘密访问密钥的值，或选择 Download .csv file（下载 .csv 文件）。这是您保存秘密访问密钥的唯一机会。将秘密访问密钥保存在安全位置后，请选择 Done（完成）。

   为用户创建访问密钥时，默认情况下，密钥对处于活动状态，并且您可以立即使用此密钥对。此时，用户拥有两个访问密钥。

2. 更新所有应用程序和工具以使用新的访问密钥。
3. 通过查看最早的访问密钥的 Last used（上次使用）信息来确定第一个访问密钥是否仍在使用。一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。
4. 即使 Last used（上次使用）信息指示旧密钥从未使用过，我们还是建议您不要立即删除第一个访问密钥。相反，选择 Actions（操作），然后选择 Deactivate（停用）以停用第一个访问密钥。
5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可以选择重新激活第一个访问密钥。然后返回到 Step 3 并更新此应用程序以使用新的密钥。
6. 在等待一段时间以确保所有应用程序和工具均已更新后，可以删除第一个访问密钥：

   a. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

   b. 在导航窗格中，选择用户。

   c. 选择预期用户的名称，然后选择安全凭证选项卡。

   d. 在要删除的访问密钥的 Access keys（访问密钥）部分中，选择 Actions（操作），然后选择 Delete（删除）。按照对话框中的说明先 Deactivate（停用），然后确认删除。

确定需要更新或删除的访问密钥（控制台）

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。

3. 如有必要，可通过完成以下步骤来将 Access key age 列添加到用户表中：

   a. 在最右侧的表上方，选择设置图标
   ( )。

   b. 在 Manage columns (管理列) 中，选择 Access key age (访问密钥使用期限)。

   c. 选择 Close 返回到用户列表。

4. Access key age 列显示最早的活动访问密钥自创建至今经过的天数。您可以利用此信息来确定可能需要更新或删除访问密钥的用户。对于没有访问密钥的用户，该列显示 None。

更新访问密钥（Amazon CLI）

您可以通过 Amazon Command Line Interface 更新访问密钥。

在不中断应用程序的情况下更新访问密钥（Amazon CLI）

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥，后者在默认情况下将处于活动状态。运行以下命令：

   - `aws iam create-access-key`

   此时，用户拥有两个访问密钥。

2. 更新所有应用程序和工具以使用新的访问密钥。

3. 通过使用以下命令，确定第一个访问密钥是否仍在使用：

   - `aws iam get-access-key-last-used`

   一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。

4. 即使步骤 Step 3 指示旧密钥未被使用，我们也建议您不要立即删除第一个访问密钥。而是通过以下命令，将第一个访问密钥的状态更改为 `Inactive`：

   - `aws iam update-access-key`

5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可将其状态切换回 `Active` 以重新激活第一个访问密钥。然后返回到步骤 Step 2 并更新此应用程序以使用新的密钥。

6. 在等待一段时间以确保所有应用程序和工具均已更新后，您可以通过以下命令删除第一个访问密钥：

- `aws iam delete-access-key`

更新访问密钥（Amazon API）

您可以使用 Amazon API 来更新访问密钥。

在不中断应用程序的情况下更新访问密钥（Amazon API）

1. 当第一个访问密钥仍处于活动状态时，创建第二个访问密钥，后者在默认情况下将处于活动状态。调用以下操作：

   - `CreateAccessKey`

   此时，用户拥有两个访问密钥。
2. 更新所有应用程序和工具以使用新的访问密钥。
3. 通过调用以下操作，确定第一个访问密钥是否仍在使用：

   - `GetAccessKeyLastUsed`

   一种方法是等待几天，然后检查旧访问密钥是否被使用，然后再继续。
4. 即使步骤 Step 3 指示旧密钥未被使用，我们也建议您不要立即删除第一个访问密钥。而是通过调用以下操作，将第一个访问密钥的状态更改为 `Inactive`：

   - `UpdateAccessKey`
5. 仅使用新的访问密钥，以确认您的应用程序可以正常工作。此时，任何仍在使用初始访问密钥的应用程序和工具将停止工作，因为它们不再具有对 Amazon 资源的访问权限。如果您发现此类应用程序或工具，可将其状态切换回 `Active` 以重新激活第一个访问密钥。然后返回到步骤 Step 2 并更新此应用程序以使用新的密钥。
6. 在等待一段时间以确保所有应用程序和工具均已更新后，您可以通过调用以下操作，删除第一个访问密钥：

   - `DeleteAccessKey`

保护访问密钥

拥有您的访问密钥的任何人将与您拥有相同的 Amazon 资源访问权限级别。因此，Amazon 全力保护您的访问密钥并确保符合我们的分担责任模型，您也应当如此。

展开以下各节以获取有助于您保护访问密钥的指导。

> **Note**
>
> 贵组织的安全要求和策略可能与本主题中介绍的有所不同。此处提供的建议旨在用作一般准则。

移除（或不生成）Amazon Web Services 账户根用户访问密钥

保护账户的最佳方法之一是不为 Amazon Web Services 账户根用户设置访问密钥。除非必须具有根用户访问密钥（这种情况很少见），否则建议不要生成根用户访问密钥。而应在 Amazon IAM Identity Center 中创建一个管理用户来执行日常管理任务。有关如何在 IAM Identity Center 中创建管理用户的信息，请参阅《IAM Identity Center 用户指南》中的 。

如果您已经拥有账户的根用户访问密钥，建议您执行以下操作：找到您当前在应用程序中使用访问密钥的位置（如果有），然后使用 IAM 用户访问密钥替换根用户访问密钥。之后再禁用并移除根用户访问密钥。有关如何更新访问密钥的更多信息，请参阅 更新访问密钥

使用临时安全凭证（IAM 角色）代替长期访问密钥

在许多情况下，您并不需要永不过期的长期访问密钥（如 IAM 用户访问密钥）。相反，您可以创建 IAM 角色并生成临时安全凭证。临时安全证书包括访问密钥 ID 和秘密访问密钥，以及一个指示证书何时到期的安全令牌。

长期访问密钥在被手动撤销之前将持续有效，例如与 IAM 用户和根用户相关的访问密钥。但是，通过 IAM 角色获取的临时安全凭证和 Amazon Security Token Service 的其他功能将在短时间内过期。凭证意外泄漏时，使用临时安全凭证可帮助降低您的风险。

在以下这些情况下使用 IAM 角色和临时安全凭证：

- 您在 Amazon EC2 实例上运行一个应用程序或 Amazon CLI 脚本。请勿直接在应用程序中使用访问密钥。请勿采取以下做法：将访问密钥传递给应用程序、将访问密钥嵌入到应用程序中、让应用程序从任何源读取密钥。相反，请定义一个对您的应用程序具有适当权限的 IAM 角色，并使用 EC2 角色启动 Amazon Elastic Compute Cloud (Amazon EC2) 实例。执行此操作会将 IAM 角色与 Amazon EC2 实例相关联。利用这种做法，还可让应用程序获取临时安全凭证，然后再使用临时凭证以编程方式调用 Amazon。Amazon 软件开发工具包和 Amazon Command Line Interface (Amazon CLI) 可以自动获得角色的临时证书。

- 您需要授予跨账户访问权限。使用 IAM 角色建立账户之间的信任，然后向用户授予有限的账户权限来访问可信账户。有关更多信息，请参阅 IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限。
- 您拥有一个移动应用程序。请勿将访问密钥嵌入应用程序，即使是嵌入加密存储也不允许。而应使用 Amazon Cognito 管理应用程序中的用户身份。此服务让您可以使用 Login with Amazon、Facebook、Google 或任何与 OpenID Connect（OIDC）兼容的身份提供商进行用户身份验证。然后，您可以使用 Amazon Cognito 凭证提供程序来管理应用程序用于向 Amazon 发出请求的凭证。
- 您希望向 Amazon 进行联合身份验证且贵组织支持 SAML 2.0。如果您所在的组织具有支持 SAML 2.0 的身份提供程序，请将提供程序配置为使用 SAML。您可以使用 SAML 与 Amazon 交换身份验证信息，并获得一组临时安全证书。有关更多信息，请参阅 SAML 2.0 联合身份验证。
- 您希望向 Amazon 进行联合身份验证且贵组织拥有本地身份存储。如果用户可以在组织内部进行身份验证，您可以编写一个可向他们颁发用于访问 Amazon 资源的临时安全凭证的应用程序。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。
- 使用 IAM 策略中的条件仅允许来自预期网络的访问。您可以通过实施具有条件的 IAM 策略来限制访问密钥的使用位置和方式，这些条件指定并仅允许预期的网络，例如您的公有 IP 地址或虚拟私有云（VPC）。这样，您就知道访问密钥只能在预期和可接受的网络中使用。

> **Note**
>
> 您是否在将 Amazon EC2 实例与需要以编程方式访问 Amazon 资源的应用程序结合使用？ 如果是，请使用 EC2 的 IAM 角色。

正确管理 IAM 用户访问密钥

如果您必须创建访问密钥才能以编程方式访问 Amazon，则应创建 IAM 用户的访问密钥，并仅向用户授予需要的权限。

请遵循以下预防措施来帮助保护 IAM 用户访问密钥：

- 请勿直接将访问密钥嵌入到代码。利用 Amazon SDK 和 Amazon 命令行工具，您可以将访问密钥放在已知位置，从而不必将其保留在代码中。

  在以下任一位置中放置访问密钥：

  - Amazon 凭证文件。Amazon 开发工具包和 Amazon CLI 自动使用您存储在 Amazon 凭证文件中的凭证。

有关使用 Amazon 证书文件的信息，请参阅软件开发工具包文档。示例包括：适用于 Java 的 Amazon SDK 开发人员指南中的设置 Amazon 凭证和区域以及 Amazon Command Line Interface 用户指南中的配置和凭证文件。

要存储适用于 适用于 .NET 的 Amazon SDK 和 Amazon Tools for Windows PowerShell 的凭证，建议您使用 SDK Store。有关更多信息，请参阅《适用于 .NET 的 Amazon SDK 开发人员指南》中的使用 SDK 存储。

- 环境变量。在多租户系统上，选择用户环境变量，而不是系统环境变量。

  有关使用环境变量存储凭证的更多信息，请参阅《Amazon Command Line Interface 用户指南》中的环境变量。

- 对不同应用程序使用不同的访问密钥。这样可以在访问密钥泄露时隔离权限并撤销单个应用程序的访问密钥。为不同的应用程序设置不同的访问密钥也会在 Amazon CloudTrail 日志文件中生成不同的条目。通过此配置，您可以更轻松地确定哪个应用程序执行了特定的操作。
- 在需要时更新访问密钥。如果存在访问密钥泄露的风险，请更新访问密钥并删除之前的访问密钥。有关详细信息，请参阅 更新访问密钥
- 删除未使用的访问密钥。如果某个用户离开了贵组织，请删除相应的 IAM 用户，以使该用户无法再访问您的资源。要找出上次使用访问密钥的时间，请使用 `GetAccessKeyLastUsed` API（Amazon CLI 命令：`aws iam get-access-key-last-used`）。
- 使用临时凭证并为最敏感的 API 操作配置多重身份验证。利用 IAM policy，可以指定用户可调用哪些 API 操作。在某些情况下，建议要求用户使用 Amazon MFA 进行身份验证，然后才允许其执行特别敏感的操作，从而提高安全性。例如，您可能拥有允许用户执行 Amazon EC2 `RunInstances`、`DescribeInstances` 和 `StopInstances` 操作的策略。但您可能希望限制破坏性操作（如 `TerminateInstances`），并确保用户只能在使用 Amazon MFA 设备进行身份验证后执行该操作。有关更多信息，请参阅 使用 MFA 保护 API 访问。

使用 Amazon 访问密钥访问移动应用程序

您可以使用 Amazon 移动应用程序访问一组有限的 Amazon 服务和功能。该移动应用程序可帮助您在外出时支持事件响应。如需了解更多信息和下载应用程序，请参阅 Amazon Console Mobile Application。

您可以使用控制台密码或访问密钥登录移动应用程序。作为最佳实践，不建议使用根用户访问密钥。相反，除在移动设备上使用密码或生物识别锁外，我们强烈建议您还应使用移动应用程序创建一个专门的 IAM 用户来管理 Amazon 资源。如果您的移动设备丢失了，您可以删除 IAM 用户的访问权限。

使用访问密钥登录（移动应用程序）

1. 在移动设备上打开该应用程序。
2. 如果这是您第一次向设备添加身份，请选择 Add an identity (添加身份)，然后选择 Access keys (访问密钥)。

   如果您已使用其他身份登录，请选择菜单图标并选择 Switch identity (切换身份)。然后选择 Sign in as a different identity (以其他身份登录)，然后选择 Access keys (访问密钥)。
3. 在 Access keys (访问密钥) 页面上输入您的信息。

   - Access key ID (访问密钥 ID) – 输入您的访问密钥 ID。
   - Secret access key (秘密访问密钥) – 输入您的秘密访问密钥。
   - Identity name (身份名称) – 输入将在移动应用程序中显示的身份名称。此名称不需要与您的 IAM 用户名一致。
   - Identity PIN (身份 PIN) – 创建将来在登录时使用的个人身份识别码（PIN）。

   > **Note**
   >
   > 如果您为 Amazon 移动应用程序启用了生物识别技术，系统将提示您使用指纹或面部识别（而非 PIN）进行验证。如果生物识别失败，系统可能会提示您输入 PIN。
4. 选择 Verify and add keys (验证并添加密钥)。

   现在，您就可以使用移动应用程序访问一组选定的资源。

相关信息

以下主题提供了有关设置 Amazon SDK 和 Amazon CLI 以使用访问密钥的指导：

- 适用于 Java 的 Amazon SDK 开发人员指南中的设置 Amazon 凭证和区域
- 适用于 .NET 的 Amazon SDK 开发人员指南中的使用 SDK Store
- 适用于 PHP 的 Amazon SDK 开发人员指南中的为 SDK 提供凭证
- Boto 3（Amazon SDK for Python）文档中的配置
- Amazon Tools for Windows PowerShell 用户指南中的使用 Amazon 凭证
- Amazon Command Line Interface 用户指南中的配置和凭证文件
- 适用于 .NET 的 Amazon SDK 开发人员指南中的使用 IAM 角色授予访问权限

- 在 Amazon SDK for Java 2.x 中为 Amazon EC2 配置 IAM 角色

使用访问密钥和私有密钥凭证访问控制台

可以使用访问密钥和私有密钥凭证进行直接访问 Amazon Web Services 管理控制台，而不仅限于 Amazon CLI。可以通过 Amazon STS `GetFederationToken` API 调用来实现这一目的。只需使用 `GetFederationToken` 提供的临时凭证和令牌构建一个控制台 URL，即可让 IAM 主体访问控制台。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。

应注意，在启用 MFA 的情况下使用 IAM 或根用户凭证直接登录控制台时，将必须使用 MFA。而使用上述方法（将临时凭证与 `GetFederationToken` 结合使用）时，则不需要 MFA。

审计访问密钥

您可以在代码中查看 Amazon 访问密钥，以确定密钥是否来自于您拥有的账户。您可以使用 `aws sts get-access-key-info` Amazon CLI 命令或 `GetAccessKeyInfo` Amazon API 操作传递访问密钥 ID。

Amazon CLI 和 Amazon API 操作返回访问密钥所属的 Amazon Web Services 账户 的 ID。以 `AKIA` 开头的访问密钥 ID 是 IAM 用户或 Amazon Web Services 账户根用户 的长期凭证。以 `ASIA` 开头的访问密钥 ID 是使用 Amazon STS 操作创建的临时凭证。如果响应中的账户属于您，则您可以根用户的身份登录并查看您的根用户访问密钥。然后，您可以提取凭证报告以了解哪个 IAM 用户拥有这些密钥。要了解谁请求了 `ASIA` 访问密钥的临时凭证，请查看 CloudTrail 日志中的 Amazon STS 事件。

为了安全起见，您可以查看 Amazon CloudTrail 日志以了解已在 Amazon 中执行操作的人员。您可以在角色信任策略中使用 `sts:SourceIdentity` 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅 `sts:SourceIdentity`。

该操作不指示访问密钥的状态。密钥可能处于活动状态、非活动状态或已删除状态 非活动密钥可能没有执行操作的权限。提供删除的访问密钥可能会返回“密钥不存在”错误。

IAM 中的 Amazon 多重身份验证

为增强安全性，我们建议您配置 Multi-Factor Authentication (MFA) 以帮助保护 Amazon 资源。您可以为所有 Amazon Web Services 账户 的 Amazon Web Services 账户根用户（包括独立账户、管理账户和成员账户）以及您的 IAM 用户启用 MFA。我们建议您尽可能使用防网络钓鱼的 MFA，例如通行密钥和安全密钥。这些基于 Fido 的身份验证器使用公钥加密技术，可抵御网络钓鱼、中间人和重放攻击，比基于 TOTP 的选项具有更强的安全级别。

对于根用户的所有账户类型，均强制实施 MFA。有关更多信息，请参阅 保护您的 Amazon Organizations 账户的根用户凭证。

当您为根用户启用 MFA 时，这只会影响根用户凭证。账户中的 IAM 用户是具有自己的凭证的不同身份，并且每个身份具有自己的 MFA 配置。有关使用 MFA 保护根用户的更多信息，请参阅 Amazon Web Services 账户根用户 的多重身份验证。

您的 Amazon Web Services 账户根用户 和 IAM 用户最多可以注册 8 台任何类型的 MFA 设备。注册多台 MFA 设备可以提供灵活性，帮助您降低设备丢失或损坏时访问中断的风险。只需一台 MFA 设备即可登录 Amazon Web Services 管理控制台 或通过 Amazon CLI 创建会话。

> **Note**
>
> 我们建议您要求您的人类用户在访问 Amazon 时使用临时凭证。您是否考虑过使用 Amazon IAM Identity Center？ 您可以使用 IAM Identity Center 集中管理对多个 Amazon Web Services 账户 的访问权限，并为用户提供受 MFA 保护的单点登录访问权限，可从一个位置访问其分配的所有账户。借助 IAM Identity Center，您可以在 IAM Identity Center 中创建和管理用户身份，或者轻松连接到现有的 SAML 2.0 兼容身份提供者。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

MFA 增加了额外安全性，要求用户在访问 Amazon 网站或服务时除了提供登录凭证之外，还需提供 Amazon 支持的 MFA 机制的唯一身份验证。

MFA 类型

Amazon 支持以下 MFA 类型：

目录

- 通行密钥和安全密钥
- 虚拟身份验证器应用程序
- 硬件 TOTP 令牌

通行密钥和安全密钥

Amazon Identity and Access Management 对 MFA 支持通行密钥和安全密钥。基于 FIDO 标准，通行密钥使用公有密钥加密技术来提供比密码更安全的强大防网络钓鱼身份验证。Amazon 支持两种类型的通行密钥：设备绑定通行密钥（安全密钥）和同步通行密钥。

- 安全密钥：这些是物理设备，例如 YubiKey，用作身份验证的第二个因素。单个安全密钥可以支持多个根用户账户和 IAM 用户。
- 同步通行密钥：它们使用来自 Google、Apple、Microsoft 账户等提供商的凭证管理器以及第三方服务（例如 1Password、Dashlane 和 Bitwarden）作为第二个因素。

您可以使用内置生物识别身份验证器（例如，Apple MacBook 上的 Touch ID）来解锁凭证管理器并登录 Amazon。通行密钥是通过您选择的提供商使用您的指纹、面部或设备 PIN 创建的。您还可以使用一台设备（如移动设备或硬件安全密钥）上的跨设备身份验证 (CDA) 密钥在另一台设备（如笔记本电脑）上登录。有关更多信息，请参阅跨设备身份验证 (CDA)。

您可以跨设备同步通行密钥以方便登录 Amazon，从而增强可用性和可恢复性。有关启用通行密钥和安全密钥的更多信息，请参阅 为根用户启用通行密钥或安全密钥（控制台）。

FIDO 联盟维护一份与 FIDO 规范兼容的所有经 FIDO 认证产品的列表。

虚拟身份验证器应用程序

虚拟身份验证器应用程序在电话或其他设备上运行，并模拟物理设备。虚拟身份验证器应用程序采用基于时间的一次性密码（TOTP）算法，并支持单个设备上的多个令牌。在登录期间，用户必须在出现提示时从该设备键入有效代码。分配给用户的每个令牌必须是唯一的。用户无法从另一个用户的令牌键入代码来进行身份验证。

我们建议您使用防网络钓鱼的 MFA，例如通行密钥或安全密钥，以获得最强的保护。如果您还不能使用通行密钥或安全密钥，我们建议您在等待硬件购买批准或等待硬件到达时，使用虚拟 MFA 设备作为临时措施。有关可用作虚拟 MFA 设备的一些受支持应用程序的列表，请参阅多重身份验证（MFA）。

有关为 IAM 用户设置虚拟 MFA 设备的说明，请参阅 在 Amazon Web Services 管理控制台 中分配虚拟 MFA 设备。

> **Note**
>
> 当您通过 Amazon Web Services 管理控制台或在登录过程中添加新的虚拟 MFA 设备时，系统会删除您的 Amazon Web Services 账户中未分配的虚拟 MFA 设备。未分配的虚拟 MFA 设备是指您账户中的设备，但在登录过程中不被账户根用户或 IAM 用户使用。已删除这些设备，因此可以向您的账户添加新的虚拟 MFA 设备。它还允许您重复使用设备名称。
>
> - 要查看账户中未分配的虚拟 MFA 设备，您可以使用 list-virtual-mfa-devices Amazon CLI 命令或 API call 调用。

- 要停用虚拟 MFA 设备，您可以使用 deactivate-mfa-device Amazon CLI 命令或 API 调用。设备将变为未分配状态。
- 要将未分配的虚拟 MFA 设备附加到您的 Amazon Web Services 账户根用户或 IAM 用户，您需要设备生成的身份验证码，以及 enable-mfa-device Amazon CLI 命令或 API 调用。

硬件 TOTP 令牌

硬件设备以基于时间的一次性密码（TOTP）算法为基础生成六位数字代码。在登录时，用户必须在另一个网页上键入来自该设备的有效代码。

这些令牌仅与 Amazon Web Services 账户 一起使用。您只能使用具有与 Amazon 安全共享的唯一令牌种子的令牌。令牌种子是在制作令牌时生成的私有密钥。从其他来源购买的令牌不能用于 IAM。为确保兼容性，您必须通过以下链接之一购买硬件 MFA 设备：OTP 令牌或 OTP 显卡。

- 分配给用户的每台 MFA 设备必须是唯一的。用户无法从另一个用户的设备键入代码来进行身份验证。有关受支持硬件 MFA 设备的信息，请参阅多重身份验证（MFA）。
- 如果想使用物理 MFA 设备，我们建议使用安全密钥来代替硬件 TOTP 设备。安全密钥不需要电池，具有防网络钓鱼功能，并可在一台设备上支持多个用户。

您只能从 Amazon Web Services 管理控制台 启用密钥或安全密钥，而不能从 Amazon CLI 或 Amazon API 启用。在启用安全密钥之前，您必须对设备拥有物理访问权限。

有关为 IAM 用户设置硬件 TOTP 令牌的说明，请参阅 在 Amazon Web Services 管理控制台 中分配硬件 TOTP 令牌。

> **Note**
>
> 基于 SMS 短信的 MFA – Amazon 已终止对短信多重身份验证（MFA）的支持。我们建议拥有使用基于短信的 MFA 的 IAM 用户的客户切换到以下替代方法之一：密码或安全密钥、虚拟（基于软件）MFA 设备或硬件 MFA 设备。您可以确定账户中拥有已分配短信 MFA 设备的用户。在 IAM 控制台中，从导航窗格中选择用户，然后在表的 MFA 列中查找具有 SMS 的用户。

MFA 建议

为了帮助保护您的 Amazon 身份，请遵循以下 MFA 身份验证建议。

- 我们建议您使用防网络钓鱼的 MFA（例如通行密钥和安全密钥 ）作为 MFA 设备。这些基于 Fido 的身份验证器可针对网络钓鱼等攻击提供最强大的保护。
- 我们建议您在您的 Amazon Web Services 账户 中为 Amazon Web Services 账户根用户 和 IAM 用户启用多台 MFA 设备。这可以使您提高 Amazon Web Services 账户 中的安全门槛，简化对高权限用户（例如 Amazon Web Services 账户根用户）的访问管理。
- 您最多可以向 Amazon Web Services 账户根用户 和 IAM 用户注册 8 台当前支持的 MFA 类型任意组合的 MFA 设备。注册多台 MFA 设备后，只需一台 MFA 设备即可以该用户的身份登录 Amazon Web Services 管理控制台 或通过 Amazon CLI 创建会话。IAM 用户必须使用现有的 MFA 设备进行身份验证，然后才能启用或禁用其他 MFA 设备。
- 如果 MFA 设备丢失、被盗或无法访问，您可以使用剩余 MFA 设备中的一台访问 Amazon Web Services 账户，而无需执行 Amazon Web Services 账户 恢复程序。如果 MFA 设备遗失或被盗，应将解除该设备与所关联 IAM 主体的关联。
- 通过使用多个 MFA，在地理位置上分散或进行远程办公的员工可以使用基于硬件的 MFA 访问 Amazon，同时无需在员工之间协调单个硬件设备的实物交换。
- 通过为 IAM 主体使用额外的 MFA 设备，您使用一个或多个 MFA 来满足日常需要，同时在安全的物理位置（例如保管库或保险柜）保存物理 MFA 设备，以满足备份和冗余的需要。

> **备注**
>
> - 您无法将安全密钥或通行密钥的 MFA 信息传递给 Amazon STS API 操作来请求临时凭证。使用安全密钥或通行密钥时，您可通过运行 `aws login` 命令获取与 Amazon CLI 和 Amazon SDK 配合使用的凭证。
> - 您无法使用 Amazon CLI 命令或 Amazon API 操作来启用 FIDO 安全密钥。
> - 多个根用户或 IAM MFA 设备不能使用相同的名称。

其他资源

以下资源可帮助您了解有关 MFA 的更多信息。

- 有关使用 MFA 访问 Amazon 的更多信息，请参阅 已启用 MFA 的登录。
- 您可以利用 IAM Identity Center 来启用对 Amazon 访问门户、IAM Identity Center 集成应用以及 Amazon CLI 的安全访问。有关更多信息，请参阅在 IAM Identity Center 中启用 MFA。

在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥

密钥是一种多重身份验证 (MFA) 设备，可用于保护您的 Amazon 资源。Amazon 支持同步密钥和设备绑定密钥（也称为安全密钥）。

同步密钥允许 IAM 用户在其许多设备（甚至是新设备）上访问其 FIDO 登录凭证，而不必在每个账户上重新注册每台设备。同步密钥包括 Google、Apple 和 Microsoft 等第一方凭证管理器以及第三方凭证管理器（例如1 Password、Dashlane 和 Bitwarden）作为第二个因素。还可以使用设备上的生物识别技术（如 TouchID、FaceID）来解锁选择的凭证管理器以使用通行密钥。

或者，设备绑定密钥绑定到 FIDO 安全密钥，您可以其插入计算机的 USB 端口，然后在出现提示时点击以安全地完成登录过程。如果您已将 FIDO 安全密钥用于其他服务，它将有一个 Amazon 支持的配置（例如，来自 Yubico 的 YubiKey 5 Series），您也可以将其用于 Amazon。否则，如果要在 Amazon 中使用面向 MFA 的 WebAuthn，您需要购买 FIDO 安全密钥。此外，FIDO 安全密钥可以在同一台设备上支持多个 IAM 用户后根用户，从而增强在保护账户安全方面的实用性。有关这两种设备类型的规格和购买信息，请参阅多重身份验证。

您最多可以向 Amazon Web Services 账户根用户 和 IAM 用户注册 8 台当前支持的 MFA 类型任意组合的 MFA 设备。注册多台 MFA 设备后，只需一台 MFA 设备即可以该用户的身份登录 Amazon Web Services 管理控制台 或通过 Amazon CLI 创建会话。建议注册多个 MFA 设备。例如，您可以注册内置身份验证器，也可以注册保存在物理安全位置的安全密钥。如果您无法使用内置身份验证器，则可以使用已注册的安全密钥。对于身份验证器应用程序，我们还建议您在这些应用程序中启用云备份或同步功能，以避免在设备丢失或损坏身份验证器应用程序时失去对账户的访问权限。

> **Note**
>
> 我们建议您要求您的人类用户在访问 Amazon 时使用临时凭证。您的用户可以与一个身份提供商联合身份到 Amazon中，使用其企业凭证和 MFA 配置进行身份验证。为了管理对 Amazon 的访问和业务应用程序，我们建议您使用 IAM Identity Center。有关更多信息，请参阅《IAM Identity Center 用户指南》。
>
> 中国区域仅支持虚拟 MFA。

主题

- 所需权限
- 为您自己的 IAM 用户启用通行密钥或安全密钥（控制台）
- 为其他 IAM 用户启用通行密钥或安全密钥（控制台）

- 替换通行密钥或安全密钥
- 使用密钥或安全密钥的受支持配置

所需权限

要为您自己的 IAM 用户管理 FIDO 密钥，同时保护与 MFA 相关的敏感操作，您必须具有以下策略中的权限：

> **Note**
>
> ARN 值是静态值，不用于指示注册身份验证器所使用的协议。我们已弃用 U2F，因此所有新的实现都使用 WebAuthn。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
```

```
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

为您自己的 IAM 用户启用通行密钥或安全密钥（控制台）

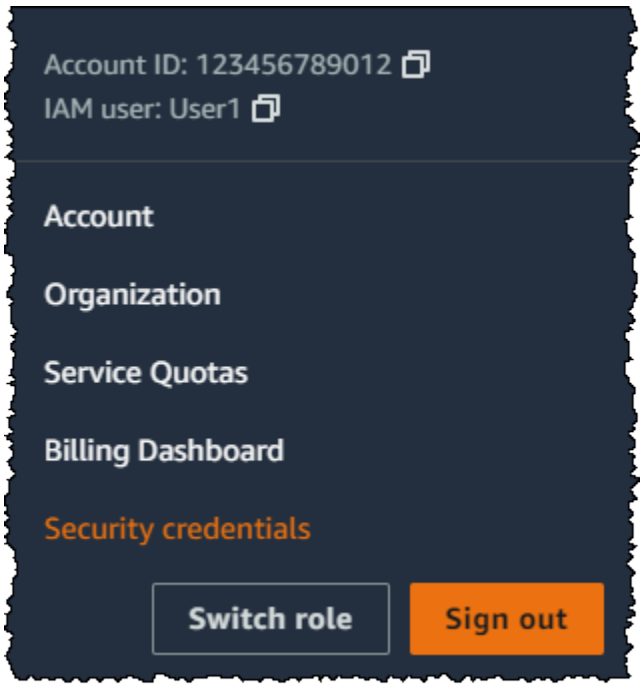
您只能从 Amazon Web Services 管理控制台 为您自己的 IAM 用户启用通行密钥或安全密钥，而不能从 Amazon CLI 或 Amazon API 启用。在启用安全密钥之前，您必须对设备拥有物理访问权限。

为您自己的 IAM 用户启用通行密钥或安全密钥（控制台）

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

   > **Note**
   >
   > 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

   要获取 Amazon Web Services 账户 ID，请联系管理员。
2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在所选 IAM 用户的页面上，选择安全凭证选项卡。
4. 在 多重身份验证 (MFA) 部分中，选择分配 MFA 设备。
5. 在 MFA 设备名称页面上，输入设备名称，选择通行密钥或安全密钥，然后选择下一步。
6. 在设置设备上，设置您的通行密钥。使用面部或指纹等生物识别数据、设备 PIN 码或将 FIDO 安全密钥插入计算机的 USB 端口并点按即可创建通行密钥。
7. 按照浏览器上的说明进行操作，然后选择继续。

现在，您已经注册了用于 Amazon 的通行密钥或安全密钥。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

为其他 IAM 用户启用通行密钥或安全密钥（控制台）

您只能从 Amazon Web Services 管理控制台 为其他 IAM 用户启用通行密钥或安全密钥，而不能从 Amazon CLI 或 Amazon API 启用。

为其他 IAM 用户启用通行密钥或安全密钥（控制台）

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 在用户下，选择要为其启用 MFA 的用户的名称。

4. 在所选 IAM 用户页面上，选择安全凭证选项卡。
5. 在 多重身份验证 (MFA) 部分中，选择分配 MFA 设备。
6. 在 MFA 设备名称页面上，输入设备名称，选择通行密钥或安全密钥，然后选择下一步。
7. 在设置设备上，设置您的通行密钥。使用面部或指纹等生物识别数据、设备 PIN 码或将 FIDO 安全密钥插入计算机的 USB 端口并点按即可创建通行密钥。
8. 按照浏览器上的说明进行操作，然后选择继续。

现在，您已经注册了供其他 IAM 用户用于 Amazon 的通行密钥或安全密钥。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

替换通行密钥或安全密钥

您一次最多可以向 Amazon Web Services 账户根用户 和 IAM 用户分配 8 台 当前支持的 MFA 类型任意组合的 MFA 设备。如果用户丢失 FIDO 身份验证器或者出于某种原因需要进行替换，必须先停用旧的 FIDO 身份验证器。然后，您可以为用户添加新 MFA 设备。

- 要停用当前与 IAM 用户关联的设备，请参阅 停用 MFA 设备。
- 要为 IAM 用户添加新的 FIDO 安全密钥，请参阅 为您自己的 IAM 用户启用通行密钥或安全密钥（控制台）。

如果无权访问新的通行密钥或安全密钥，则可以启用新的虚拟 MFA 设备或硬件 TOTP 令牌。有关说明，请参阅以下章节之一：

- 在 Amazon Web Services 管理控制台 中分配虚拟 MFA 设备
- 在 Amazon Web Services 管理控制台 中分配硬件 TOTP 令牌

使用密钥或安全密钥的受支持配置

您可以通过使用当前支持的配置，将 FIDO2 设备绑定密钥（也称为安全密钥）配置为 IAM 的多重身份验证（MFA）方法。这包括 IAM 支持的 FIDO2 设备及支持 FIDO2 的浏览器。在注册 FIDO2 设备之前，请确认您使用的是最新版本的浏览器和操作系统（OS）。相关功能在不同的浏览器、身份验证器和操作系统客户端上可能有不同的行为。如果您的设备在一种浏览器上注册失败，则可以尝试使用其他浏览器注册。

FIDO2 是开放身份验证标准，作为 FIDO U2F 的扩展，基于公有密钥加密提供同样高级别的安全性。FIDO2 由 W3C Web 身份验证规范（WebAuthn API）和 FIDO 联盟客户端到身份验证器协议（CTAP，一种应用程序层协议）组成。CTAP 支持客户端或平台（如浏览器或操作系统）与外部身份

验证器之间进行通信。当您在 Amazon 中启用经 FIDO 认证的身份验证器时，安全密钥会创建仅适用于 Amazon 的新密钥对。首先，输入您的凭证。出现提示时，点击安全密钥，这会响应 Amazon 发出的身份验证质询。要了解有关 FIDO2 标准的更多信息，请参阅 FIDO2 项目。

Amazon 支持的 FIDO2 设备

IAM 支持 FIDO2 安全设备，这些设备可通过 USB、Bluetooth或 NFC 连接到您的设备。IAM 还支持 TouchID 或 FaceID 等平台身份验证器。IAM 不支持 Windows Hello 的本地密钥注册。若要创建和使用密钥，Windows 用户应使用跨设备身份验证，即使用来自一台设备（例如移动设备）的密钥或硬件安全密钥在另一台设备（如笔记本电脑）上登录。

> **Note**
>
> Amazon 需要访问您的计算机上的物理 USB 端口以验证您的 FIDO2 设备。安全密钥不支持虚拟机、远程连接或浏览器的无痕模式。

FIDO 联盟维护一份与 FIDO 规范兼容的所有 FIDO2 产品的列表。

支持 FIDO2 的浏览器

FIDO2 安全设备能否在 Web 浏览器中运行，取决于具体的浏览器和操作系统组合。以下浏览器当前支持使用安全密钥：

Web 浏览器	macOS 10.15+	Windows 10	Linux	iOS 14.5+	Android 7+
Chrome	支持	是	是	是	否
Safari	是	否	否	是	否
Edge	是	是	否	是	否
Firefox	支持	是	否	是	否

> **Note**
>
> 当前支持 FIDO2 的大多数 Firefox 版本默认情况下不会启用支持。有关在 Firefox 中启用 FIDO2 支持的说明，请参阅 排查通行密钥和 FIDO 安全密钥问题。

macOS 上的 Firefox 可能不完全支持通行密钥的跨设备身份验证工作流程。系统可能会提示您触摸安全密钥，而不是继续进行跨设备身份验证。建议使用其他浏览器（例如 Chrome 或 Safari）在 macOS 上使用通行密钥登录。

要详细了解支持 FIDO2 认证设备（如 YubiKey）的浏览器，请参阅 Operating system and web browser support for FIDO2 and U2F。

浏览器插件

Amazon 仅支持原生支持 FIDO2 的浏览器。Amazon 不支持使用插件来添加 FIDO2 浏览器支持。某些浏览器插件与 FIDO2 标准不兼容，这可能会导致 FIDO2 安全密钥产生意外结果。

有关禁用浏览器插件和其他问题排查提示的信息，请参阅我无法启用我的 FIDO 安全密钥。

设备认证

我们仅在注册安全密钥期间捕获和分配与设备相关的认证，例如 FIPS 验证和 FIDO 认证级别。从 FIDO Alliance Metadata Service（MDS）获取设备认证。如果您的安全密钥的认证状态或级别发生变化，则不会自动反映在设备标签中。要更新设备的认证信息，请再次注册设备，以获取更新的认证信息。

Amazon 在设备注册期间提供以下认证类型作为条件密钥，从 FIDO MDS 获得：
FIPS-140-2、FIPS-140-3 和 FIDO 认证级别。您可以根据首选认证类型和级别，在其 IAM policy 中指定特定身份验证器的注册。有关更多信息，请参阅下面的策略。

设备认证策略示例

以下用例显示允许您为 MFA 设备注册 FIPS 认证的示例策略。

主题

- 用例 1：只允许注册通过 FIPS-140-2 L2 认证的设备
- 用例 2：允许注册通过 FIPS-140-2 L2 和 FIDO L1 认证的设备
- 用例 3：允许注册通过 FIPS-140-2 L2 或 FIPS-140-3 L2 认证的设备
- 使用案例 4：允许已通过 FIPS-140-2 L2 认证并支持其他 MFA 类型（例如虚拟身份验证器和硬件 TOTP）的设备注册

用例 1：只允许注册通过 FIPS-140-2 L2 认证的设备

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]
}
```

用例 2：允许注册通过 FIPS-140-2 L2 和 FIDO L1 认证的设备

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
```

```
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2",
                    "iam:FIDO-certification": "L1"
                }
            }
        }
    ]
}
```

用例 3：允许注册通过 FIPS-140-2 L2 或 FIPS-140-3 L2 认证的设备

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L2"
                }
            }
        }
    ]
}
```

使用案例 4：允许已通过 FIPS-140-2 L2 认证并支持其他 MFA 类型（例如虚拟身份验证器和硬件 TOTP）的设备注册

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:RegisterSecurityKey": "Create"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
```

```
      "Condition": {
        "StringEquals": {
          "iam:RegisterSecurityKey": "Activate",
          "iam:FIDO-FIPS-140-2-certification": "L2"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "iam:EnableMFADevice",
      "Resource": "*",
      "Condition": {
        "Null": {
          "iam:RegisterSecurityKey": "true"
        }
      }
    }
  ]
}
```

Amazon CLI 和 Amazon API

Amazon 支持仅在 Amazon Web Services 管理控制台 中使用密钥或安全密钥。在 Amazon CLI 和 Amazon API 中不支持使用密钥和安全密钥进行 MFA，也不支持使用密钥和安全密钥访问 MFA 保护的 API 操作。

其他资源

- 有关在 Amazon 中使用密钥和安全密钥的更多信息，请参阅 在 Amazon Web Services 管理控制台中分配通行密钥或安全密钥。
- 有关在 Amazon 中对密钥和安全密钥进行故障排除的帮助信息，请参阅 排查通行密钥和 FIDO 安全密钥问题。
- 有关 FIDO2 支持的一般行业信息，请参阅 FIDO2 项目。

在 Amazon Web Services 管理控制台 中分配虚拟 MFA 设备

> **Important**
>
> Amazon 建议您在登录 Amazon 时，尽可能使用通行密钥或安全密钥作为 MFA。有关更多信息，请参阅 在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥。

您可以将手机或其他设备作为虚拟多重身份验证（MFA）设备。为此，请安装符合 RFC 6238 的移动应用程序，这是一种基于标准的 TOTP（基于时间的一次性密码）算法。这些应用程序生成六位数的身份验证代码。由于身份验证器可以在不安全的移动设备上运行，而且代码有可能与未经授权的各方共享，因此基于 TOTP 的 MFA 无法提供与 FIDO2 安全密钥和通行密钥等防网络钓鱼选项相同的安全级别。我们建议您使用通行密钥或安全密钥进行 MFA，以最有效地防范网络钓鱼等攻击。

如果您还不能使用通行密钥或安全密钥，我们建议您在等待硬件购买批准或等待硬件到达时，使用虚拟 MFA 设备作为临时措施。

大多数虚拟 MFA 应用程序支持创建多个虚拟设备，从而允许您在多个 Amazon Web Services 账户 或用户中使用相同的应用程序。您最多可以向 Amazon Web Services 账户根用户 和 IAM 用户注册 8 台 MFA 类型任意组合的 MFA 设备。只需一台 MFA 设备即可登录 Amazon Web Services 管理控制台 或通过 Amazon CLI 创建会话。建议注册多个 MFA 设备。对于身份验证器应用程序，我们还建议您启用云备份或同步功能，以帮助避免在设备丢失或损坏时失去对账户的访问权限。

Amazon 需要可产生六位数 OTP 的虚拟 MFA 应用程序。有关您可以使用的虚拟 MFA 应用程序的列表，请参阅 Multi-Factor Authentication。

主题

- 所需权限
- 为 IAM 用户启用虚拟 MFA 设备（控制台）
- 替换虚拟 MFA 设备

所需权限

要管理您的 IAM 用户的虚拟 MFA 设备，您必须具有以下策略中的权限：Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的 MFA 设备。。

为 IAM 用户启用虚拟 MFA 设备（控制台）

您可在 Amazon Web Services 管理控制台 中使用 IAM 为您账户中的 IAM 用户启用和管理虚拟 MFA 设备。您可以将标签附加到 IAM 资源（包括虚拟 MFA 设备），以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记虚拟 MFA 设备。要使用 Amazon CLI 或 Amazon API 启用和管理 MFA 设备，请参阅 在 Amazon CLI 或 Amazon API 中分配 MFA 设备。有关标记 IAM 资源的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。

> **Note**
>
> 您必须拥有对将托管用户的虚拟 MFA 设备的硬件的物理访问权限以便配置 MFA。例如，您可能为使用在智能手机上运行的虚拟 MFA 设备的用户配置 MFA。在这种情况下，您必须具有智能手机才能完成该向导。因此，您可能想让用户配置和管理他们自己的虚拟 MFA 设备。在此情况下，您必须授予用户执行必要的 IAM 操作所需的权限。有关更多信息以及授予这些权限的 IAM 策略示例，请参阅 IAM 教程：允许用户管理其凭证和 MFA 设置 和示例策略 Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的 MFA 设备。。

为 IAM 用户启用虚拟 MFA 设备（控制台）

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 在 用户列表中，选择 IAM 用户的名称。
4. 选择 Security Credentials (安全凭证) 选项卡。在 Multi-factor authentication (MFA) [多重身份验证（MFA）] 部分中，选择 Assign MFA device（分配 MFA 设备）。
5. 在向导中，键入设备名称，选择身份验证器应用程序，然后选择下一步。

   IAM 将生成并显示虚拟 MFA 设备的配置信息，包括 QR 代码图形。此图形是秘密配置密钥的表示形式，适用于不支持 QR 代码的设备上的手动输入。
6. 打开您的虚拟 MFA 应用程序。有关可用于托管虚拟 MFA 设备的应用程序的列表，请参阅多重身份验证。

   如果虚拟 MFA 应用程序支持多个虚拟 MFA 设备或账户，请选择相应的选项以创建新的虚拟 MFA 设备或账户。
7. 确定 MFA 应用程序是否支持 QR 代码，然后执行以下操作之一：

- 在向导中，选择 Show QR 代码 (显示 QR 代码)，然后使用该应用程序扫描 QR 代码。这可能是摄像头图标或使用设备的摄像头扫描代码的扫描代码选项。
- 在向导中，选择 Show secret key（显示私有密钥），然后在您的 MFA 应用程序中键入私有密钥。

完成操作后，虚拟 MFA 设备会开始生成一次性密码。

8. 在设置设备页面中的 MFA 代码 1 框中，键入虚拟 MFA 设备当前显示的一次性密码。请等候 30 秒，以便设备生成新的一次性密码。然后在 MFA code 2 (MFA 代码 2) 框中键入第二个一次性密码。选择 Add MFA（添加 MFA）。

> **Important**
>
> 生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

虚拟 MFA 设备现在已准备好与 Amazon 一起使用了。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

> **Note**
>
> 当您通过 Amazon Web Services 管理控制台或在登录过程中添加新的虚拟 MFA 设备时，系统会删除您 Amazon Web Services 账户中未分配的虚拟 MFA 设备。未分配的虚拟 MFA 设备是指您账户中的设备，但在登录过程中不被账户根用户或 IAM 用户使用。已删除这些设备，因此可以向您的账户添加新的虚拟 MFA 设备。它还允许您重复使用设备名称。
>
> - 要查看账户中未分配的虚拟 MFA 设备，您可以使用 list-virtual-mfa-devices Amazon CLI 命令或 API 调用。
> - 要停用虚拟 MFA 设备，您可以使用 deactivate-mfa-device Amazon CLI 命令或 API 调用。设备将变为未分配状态。
> - 要将未分配的虚拟 MFA 设备附加到您的 Amazon Web Services 账户根用户或 IAM 用户，您需要设备生成的身份验证码，以及 enable-mfa-device Amazon CLI 命令或 API 调用。

替换虚拟 MFA 设备

您的 Amazon Web Services 账户根用户 和 IAM 用户最多可以注册 8 台 MFA 类型任意组合的 MFA 设备。如果用户丢失设备或出于任何原因需要更换设备，请先停用旧设备。然后，您可以为用户添加新设备。

- 要停用当前与其他 IAM 用户关联的设备，请参阅 停用 MFA 设备。
- 要为其他 IAM 用户添加替换虚拟 MFA 设备，请按照上述 为 IAM 用户启用虚拟 MFA 设备（控制台）过程中的步骤进行操作。
- 要为 Amazon Web Services 账户根用户添加替换虚拟 MFA 设备，请按照 为根用户启用虚拟 MFA 设备（控制台）过程中的步骤操作。

在 Amazon Web Services 管理控制台 中分配硬件 TOTP 令牌

> **Important**
>
> Amazon 建议您在登录 Amazon 时，尽可能使用通行密钥或安全密钥作为 MFA。有关更多信息，请参阅 在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥。

硬件 TOTP 令牌以基于时间的一次性密码（TOTP）算法为基础生成一个六位数字代码。在登录过程中，用户必须在出现提示时从该设备键入有效代码。分配给用户的每个 MFA 设备都必须是唯一的；进行身份验证时，某个用户无法从另一个用户的设备键入代码。MFA 设备不能跨账户或用户共享。

硬件 TOTP 令牌和 FIDO 安全密钥都是您购买的物理设备。硬件 MFA 设备会在您登录 Amazon 时生成 TOTP 验证码来验证身份。此类设备需要使用电池，并且随着时间推移可能需要更换以及与 Amazon 重新同步。FIDO 安全密钥采用公钥加密技术，不需要电池，并且可提供无缝的身份验证流程。我们建议使用 FIDO 安全密钥，是因为此类密钥能够抵御网络钓鱼攻击，是 TOTP 设备的更安全替代方案。此外，FIDO 安全密钥可以在同一台设备上支持多个 IAM 用户后根用户，从而增强在保护账户安全方面的实用性。有关这两种设备类型的规格和购买信息，请参阅多重身份验证。

您可以从 Amazon Web Services 管理控制台、命令行或 IAM API 为 IAM 用户启用硬件 TOTP 令牌。要为 Amazon Web Services 账户根用户 启用 MFA 设备，请参阅为根用户启用硬件 TOTP 令牌（控制台）。

您最多可以向 Amazon Web Services 账户根用户 和 IAM 用户注册 8 台当前支持的 MFA 类型任意组合的 MFA 设备。注册多台 MFA 设备后，只需一台 MFA 设备即可以该用户的身份登录 Amazon Web Services 管理控制台 或通过 Amazon CLI 创建会话。

> **Important**
>
> 我们建议您为用户启用多台 MFA 设备，以便在 MFA 设备丢失或无法访问时能够继续访问您的账户。

> **Note**
>
> 如果想从命令行启用 MFA 设备，请使用 `aws iam enable-mfa-device`。如要使用 IAM API 启用 MFA 设备，请使用 `EnableMFADevice` 操作。

主题

- 所需权限
- 为您自己的 IAM 用户启用硬件 TOTP 令牌（控制台）
- 为其他 IAM 用户启用硬件 TOTP 令牌（控制台）
- 替换物理 MFA 设备

所需权限

要为您自己的 IAM 用户管理硬件 TOTP 令牌，同时保护与 MFA 相关的敏感操作，您必须具有以下策略中的权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
```

```
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

为您自己的 IAM 用户启用硬件 TOTP 令牌（控制台）

您可以从 Amazon Web Services 管理控制台 启用您自己的硬件 TOTP 令牌。

> **Note**
>
> 在启用硬件 TOTP 令牌之前，您必须拥有对设备的物理访问权限。

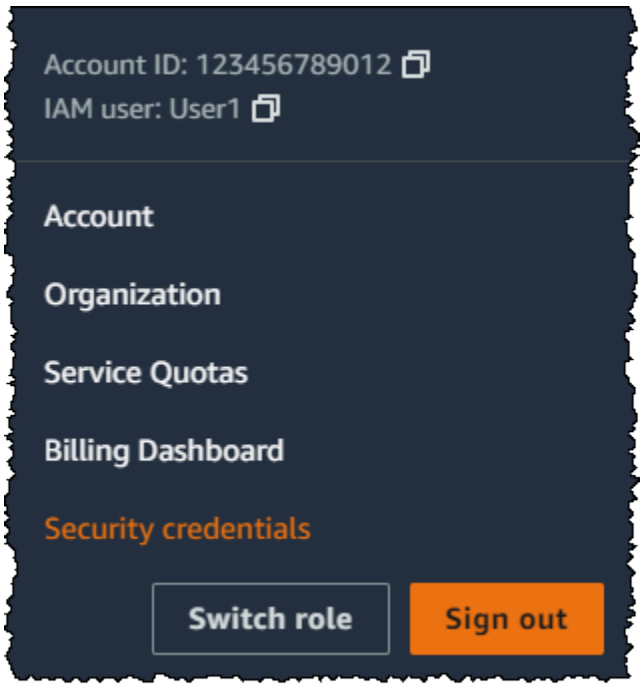
为您自己的 IAM 用户启用硬件 TOTP 令牌（控制台）

1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

> **Note**
>
> 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Amazon IAM 凭证选项卡的多重身份验证 (MFA) 部分中，选择分配 MFA 设备。
4. 在向导中，键入 Device name（设备名称），选择 Hardware TOTP token（硬件 TOTP 令牌），然后选择 Next（下一步）。
5. 键入设备序列号。序列号通常位于设备的背面。
6. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。

7. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。
8. 选择 Add MFA（添加 MFA）。

> **Important**
>
> 在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

为其他 IAM 用户启用硬件 TOTP 令牌（控制台）

您可以从 Amazon Web Services 管理控制台 为其他 IAM 用户启用硬件 TOTP 令牌。

为其他 IAM 用户启用硬件 TOTP 令牌（控制台）

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 选择要为其启用 MFA 的用户的名称。
4. 选择 Security Credentials (安全凭证) 选项卡。在多重身份验证 (MFA) 部分中，选择分配 MFA 设备。
5. 在向导中，键入 Device name（设备名称），选择 Hardware TOTP token（硬件 TOTP 令牌），然后选择 Next（下一步）。
6. 键入设备序列号。序列号通常位于设备的背面。
7. 在 MFA code 1 (MFA 代码 1) 框中，输入 MFA 设备显示的六位数编码。您需要按设备正面的按钮来显示编码。

8. 在设备刷新期间等候 30 秒，然后在 MFA code 2 (MFA 代码 2) 框中键入第二个六位数编码。您需要再次按设备正面的按钮来显示第二个编码。
9. 选择 Add MFA（添加 MFA）。

> **Important**
>
> 在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。这种情况下，您可以重新同步设备。

设备已准备就绪，可在 Amazon 上使用。有关在 Amazon Web Services 管理控制台上使用 MFA 的信息，请参阅 已启用 MFA 的登录。

替换物理 MFA 设备

您一次最多可以向您的 Amazon Web Services 账户根用户 和 IAM 用户分配 8 台当前支持的 MFA 类型任意组合的 MFA 设备。如果用户丢失设备或出于任何原因需要替换它，您必须先停用旧设备。然后，您可以为用户添加新设备。

- 要停用当前与用户关联的设备，请参阅停用 MFA 设备。
- 要为 IAM 用户添加替代用硬件 TOTP 令牌，请按照本主题前面部分的过程 为其他 IAM 用户启用硬件 TOTP 令牌（控制台） 中的步骤进行操作。
- 要为 Amazon Web Services 账户根用户 添加替代用硬件 TOTP 令牌，请按照本主题前面部分的过程 为根用户启用硬件 TOTP 令牌（控制台） 中的步骤进行操作。

在 Amazon CLI 或 Amazon API 中分配 MFA 设备

您可以使用 Amazon CLI 命令或 Amazon API 操作为 IAM 用户启用虚拟 MFA 设备。您无法使用 Amazon CLI、Amazon API、Tools for Windows PowerShell 或任何其他命令行工具为 Amazon Web Services 账户根用户 启用 MFA 设备。不过，可以使用 Amazon Web Services 管理控制台 为根用户启用 MFA 设备。

当您从 Amazon Web Services 管理控制台启用 MFA 设备时，控制台会为您执行多个步骤。如果您改用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 创建虚拟设备，则必须按正确的顺序手动执行这些步骤。例如，要创建虚拟 MFA 设备，您必须创建 IAM 对象，将代码提取为字符串或 QR 代码图形。然后，您必须同步该设备并将其与 IAM 用户关联。有关更多详细信息，请参阅 New-IAMVirtualMFADevice 的 Examples（示例）部分。对于物理设备，您可以跳过创建步骤，直接同步该设备并将其与用户关联。

您可以将标签附加到 IAM 资源（包括虚拟 MFA 设备），以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记虚拟 MFA 设备。

使用 SDK 或 CLI 的 IAM 用户可以通过调用 EnableMFADevice 来启用其他 MFA 设备，也可以通过调用 DeactivateMFADevice 来停用现有的 MFA 设备。要成功执行此操作，用户必须首先调用 GetSessionToken 并使用一个现有的 MFA 设备提交 MFA 代码。此调用将会返回临时安全凭证，然后才可以使用这些凭证对需要 MFA 身份验证的 API 操作进行签名。有关示例请求和响应，请参阅 GetSessionToken– 不可信环境中用户的临时凭证。

在 IAM 中创建虚拟设备实体来代表虚拟 MFA 设备

这些命令提供在以下很多命令中代替序列号的设备 ARN。

- Amazon CLI: aws iam create-virtual-mfa-device
- Amazon API：CreateVirtualMFADevice

启用 MFA 设备，以便在 Amazon 上使用

这些命令将设备与 Amazon 同步，并将其与用户关联。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。

> **Important**
>
> 在生成身份验证代码后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。如果发生这种情况，可以使用下面介绍的命令重新同步设备。

- Amazon CLI: aws iam enable-mfa-device
- Amazon API：EnableMFADevice

停用设备

使用这些命令可取消设备与用户的关联并停用设备。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。您还必须单独删除虚拟设备实体。

- Amazon CLI: aws iam deactivate-mfa-device
- Amazon API：DeactivateMFADevice

列出虚拟 MFA 设备实体

使用以下命令列出虚拟 MFA 设备实体。

- Amazon CLI: `aws iam list-virtual-mfa-devices`
- Amazon API：`ListVirtualMFADevices`

要标记虚拟 MFA 设备

使用这些命令来标记虚拟 MFA 设备。

- Amazon CLI: `aws iam tag-mfa-device`
- Amazon API：`TagMFADevice`

要为虚拟 MFA 设备列出标签

使用这些命令列出附加到虚拟 MFA 设备的标签。

- Amazon CLI: `aws iam list-mfa-device-tags`
- Amazon API：`ListMFADeviceTags`

要取消虚拟 MFA 设备的标签

使用这些命令删除附加到虚拟 MFA 设备的标签。

- Amazon CLI: `aws iam untag-mfa-device`
- Amazon API：`UntagMFADevice`

重新同步 MFA 设备

如果设备生成的代码不被 Amazon 接受，则使用这些命令。如果设备是虚拟设备，则将虚拟设备的 ARN 用作序列号。

- Amazon CLI: `aws iam resync-mfa-device`
- Amazon API：`ResyncMFADevice`

删除 IAM 中的虚拟 MFA 设备实体

在取消设备与用户的关联后，可以删除设备实体。

- Amazon CLI: `aws iam delete-virtual-mfa-device`
- Amazon API：`DeleteVirtualMFADevice`

恢复丢失或无法正常工作的虚拟 MFA 设备

有时，托管虚拟 MFA 应用程序的用户设备丢失、更换或无法正常工作。在这种情况下，用户无法自行将其恢复。该用户必须与管理员联系，才能将设备停用。有关更多信息，请参阅 在 IAM 中恢复受 MFA 保护的身份。

检查 MFA 状态

使用 IAM 控制台检查 Amazon Web Services 账户根用户 或 IAM 用户是否启用了有效的 MFA 设备。

检查根用户的 MFA 状态

1. 使用您的根用户凭证登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.amazonaws.cn/iam/。
2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。
3. 在 Multi-Factor Authentication (MFA) 下查看 MFA 是已启用还是已禁用。如果尚未激活 MFA，系统会显示一个提醒符号
   ( )。

如果要为账户启用 MFA，请参阅以下章节之一：

- 为根用户启用虚拟 MFA 设备（控制台）
- 为根用户启用通行密钥或安全密钥（控制台）
- 为根用户启用硬件 TOTP 令牌（控制台）

检查 IAM 用户的 MFA 状态

1. 使用 https://console.amazonaws.cn/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Users（用户）。
3. 如有必要，请通过完成以下步骤来将 MFA 列添加到用户表中：
   a. 在最右侧的表上方，选择设置图标
   ( )。

b. 在 Manage Columns 中，选择 MFA。

c. （可选）清除不希望在用户表中显示的任何列标题的复选框。

d. 选择 Close 返回到用户列表。

4. MFA 列告知您 MFA 设备的启用情况。如果用户没有处于活动状态的 MFA 设备，控制台将显示 None（无）。如果用户启用了 MFA 设备，则 MFA 列将显示启用的设备类型，并附上值 Virtual（虚拟）、Security key（安全密钥）、Hardware（硬件）或 SMS（短信）。

> **Note**
>
> Amazon 已终止对短信多重身份验证（MFA）的支持。我们建议拥有使用基于短信的 MFA 的 IAM 用户的客户切换到以下替代方法之一：虚拟（基于软件）MFA 设备、FIDO 安全密钥或硬件 MFA 设备。您可以确定账户中拥有已分配短信 MFA 设备的用户。为此，请转到 IAM 控制台，从导航窗格中选择用户，然后在表的 MFA 列中查找具有 SMS 的用户。

5. 要查看有关用户的 MFA 设备的其他信息，请选择要检查其 MFA 状态的用户的名称，然后选择 Security credentials 选项卡。

6. 如果用户没有处于活动状态的 MFA 设备，控制台将显示无 MFA 设备。在多重身份验证（MFA）部分中分配 MFA 设备以提高 Amazon 环境的安全性。如果用户启用了 MFA 设备，则 Multi-factor authentication (MFA) [多重身份验证（MFA）] 部分显示有关这些设备的详细信息：

   - 设备名称
   - 设备类型
   - 设备的标识符，例如物理设备的设备序列号或虚拟设备的 Amazon 中的 ARN
   - 何时创建设备

要删除或重新同步设备，请选择设备旁边的单选按钮，然后选择 Remove（删除）或 Resync（重新同步）。

有关启用 MFA 的更多信息，请参阅以下章节：

- 在 Amazon Web Services 管理控制台 中分配虚拟 MFA 设备
- 在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥
- 在 Amazon Web Services 管理控制台 中分配硬件 TOTP 令牌

重新同步虚拟和硬件 MFA 设备

您可以使用 Amazon 重新同步虚拟和硬件多重身份验证 (MFA) 设备。如果您尝试使用的设备并未同步，则将导致用户的登录尝试失败，IAM 会提示您重新同步设备。

> **Note**
>
> FIDO 安全密钥不同步。如果 FIDO 安全密钥丢失或损坏，您可以将其停用。有关停用任何 MFA 设备类型的说明，请参阅停用其他 IAM 用户的 MFA 设备（控制台）。

作为 Amazon 管理员，您可以重新同步您的 IAM 用户无法同步的虚拟和硬件 MFA 设备。

如果您的 Amazon Web Services 账户根用户 MFA 设备不工作，可以使用 IAM 控制台重新同步该设备（完成登录过程与否均可实现）。如果您无法成功地重新同步设备，则可能需要取消关联并重新关联该设备。有关此操作的更多信息，请参阅 停用 MFA 设备 和 IAM 中的 Amazon 多重身份验证。

主题

- 所需权限
- 重新同步虚拟和硬件 MFA 设备（IAM 控制台）
- 重新同步虚拟和硬件 MFA 设备 (Amazon CLI)
- 重新同步虚拟和硬件 MFA 设备 (Amazon API)

所需权限

要为您自己的 IAM 用户重新同步虚拟或硬件 MFA 设备，您必须具有以下策略的权限。此策略不允许您创建或停用设备。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListVirtualMFADevices"
```

```
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToViewAndManageTheirOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "BlockAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

重新同步虚拟和硬件 MFA 设备（IAM 控制台）

您可以使用 IAM 控制台重新同步虚拟和硬件 MFA 设备。

重新同步您自己 IAM 用户的虚拟或硬件 MFA 设备（控制台）

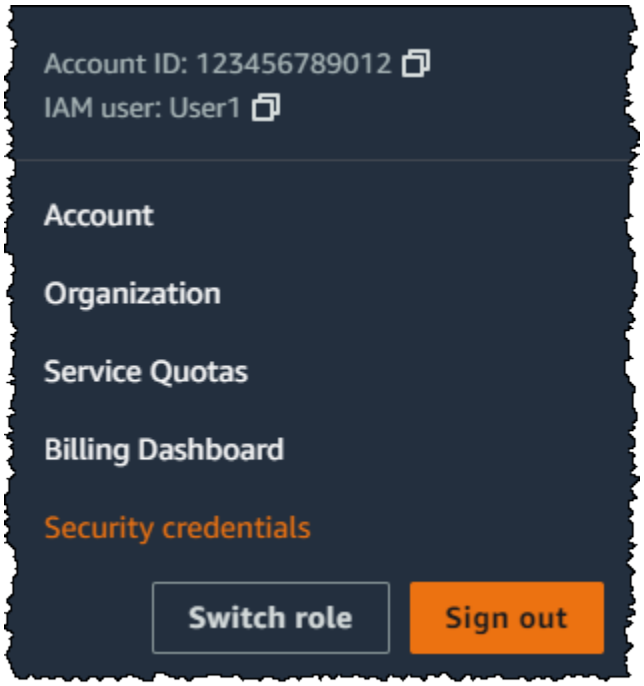
1. 使用 Amazon 账户 ID 或账户别名、您的 IAM 用户名和密码登录到 IAM 控制台。

> **Note**
>
> 为方便起见，Amazon 登录页面使用浏览器 Cookie 记住您的 IAM 用户名和账户信息。如果您之前以其他用户身份登录，请选择页面底部的 Sign-in to a different account（登录到

其他账户）以返回主登录页面。在此处，您可以输入要重新导向到您账户 IAM 用户登录页面的 Amazon 账户 ID 或账户别名。

要获取 Amazon Web Services 账户 ID，请联系管理员。

2. 在右上角的导航栏中，选择您的用户名，然后选择 Security credentials（安全凭证）。



3. 在 Amazon IAM 凭证选项卡的多重身份验证（MFA）部分中，选择 MFA 设备旁边的单选按钮，然后选择重新同步。
4. 依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Resync（重新同步）。

> **Important**
>
> 生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求看起来有效，但实际上设备仍然不同步。这是因为基于时间的一次性密码（TOTP）很快会过期。

重新同步其他 IAM 用户的虚拟或硬件 MFA 设备（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在导航窗格中，选择 Users，然后选择其 MFA 设备需要重新同步的用户的名称。
3. 选择安全凭证选项卡。在多重身份验证（MFA）部分中，选择 MFA 设备旁边的单选按钮，然后选择重新同步。
4. 依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Resync（重新同步）。

> **Important**
>
> 生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求看起来有效，但实际上设备仍然不同步。这是因为基于时间的一次性密码（TOTP）很快会过期。

在登录前重新同步您的根用户 MFA（控制台）

1. 在 Amazon Web Services Sign In With Authentication Device（亚马逊云科技使用身份验证设备登录）页面上，选择 Having problems with your authentication device?（身份验证设备出现问题？）Click here（点击此处）。

> **Note**
>
> 您可能会看到不同的文本，例如使用 MFA 登录和排除您的身份验证设备故障。不过，它们提供了相同的功能。

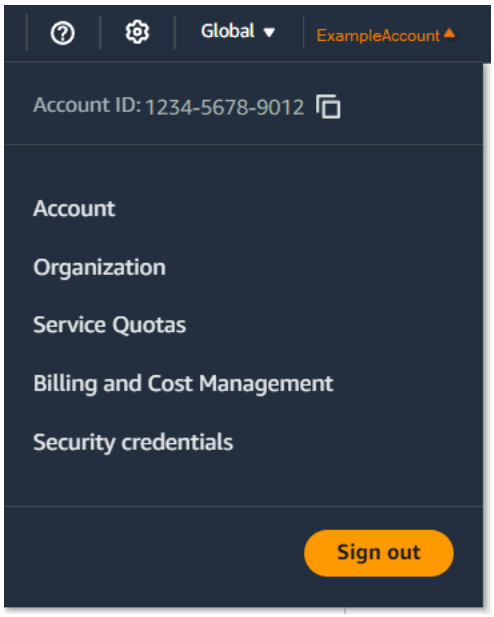
2. 在 Re-Sync With Our Servers (与服务器重新同步) 部分中，依次将设备上按顺序生成的两个代码键入 MFA code 1 (MFA 代码 1) 和 MFA code 2 (MFA 代码 2) 中。然后选择 Re-sync authentication device。
3. 如有必要，请再次键入您的密码，然后选择登录。然后使用您的 MFA 设备完成登录。

在登录后重新同步您的根用户 MFA 设备（控制台）

1. 选择 Root user（根用户）并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 IAM 控制台。在下一页上，输入您的密码。

> **Note**
>
> 作为根用户，您无法登录到以 IAM 用户身份登录页面。如果您看到以 IAM 用户身份登录页面，请选择该页面底部附近的使用根用户电子邮件登录。要获取以根用户身份登录方面的帮助，请参阅《Amazon 登录 用户指南》中的以根用户身份登录 Amazon Web Services 管理控制台。

2. 在导航栏的右侧选择您的账户名称，然后选择 Security credentials（安全凭证）。如有必要，选择 Continue to Security Credentials（继续使用安全凭证）。



3. 展开页面上的多重身份验证（MFA）(MFA) 部分。
4. 选中设备旁边的单选按钮，然后选择 Resync（重新同步）。
5. 在 Resync MFA device（重新同步 MFA 设备）对话框中，依次将设备上按顺序生成的两个代码键入 MFA code 1（MFA 代码 1）和 MFA code 2（MFA 代码 2）中。然后选择 Resync（重新同步）。

> **Important**
>
> 生成代码之后立即提交您的请求。如果生成代码后等待很长时间才提交请求，MFA 设备会成功与用户关联，但 MFA 设备无法同步。这是因为基于时间的一次性密码（TOTP）很快会过期。

重新同步虚拟和硬件 MFA 设备 (Amazon CLI)

您可以从 Amazon CLI 重新同步虚拟和硬件 MFA 设备。

重新同步 IAM 用户的虚拟或硬件 MFA 设备 (Amazon CLI)

在命令提示符处，发布 aws iam resync-mfa-device 命令：

- 虚拟 MFA 设备：将设备的 Amazon 资源名称 (ARN) 指定为序列号。

```
aws iam resync-mfa-device --user-name Richard --serial-number
 arn:aws:iam::123456789012:mfa/RichardsMFA --authentication-code1 123456 --
authentication-code2 987654
```

- 硬件 MFA 设备：将硬件设备的序列号指定为序列号。格式因供应商而异。例如，您可以从 Amazon 购买 gemalto 令牌。它的序列号通常是四个字母，后跟四个数字。

```
aws iam resync-mfa-device --user-name Richard --serial-number ABCD12345678 --
authentication-code1 123456 --authentication-code2 987654
```

> **Important**
>
> 生成代码之后立即提交您的请求。如果您生成代码，然后等待了很长时间才提交请求，则请求将失败，因为代码很快就会过期。

重新同步虚拟和硬件 MFA 设备 (Amazon API)

IAM 有一个执行同步的 API 调用。在这种情况下，建议您授予虚拟和硬件 MFA 设备用户访问此 API 调用的权限。然后，基于此 API 调用构建工具，这样您的用户即可随时根据需要重新同步其设备。

重新同步 IAM 用户的虚拟或硬件 MFA 设备 (Amazon API)

- 发送 ResyncMFADevice 请求。

停用 MFA 设备

如果您在使用多重身份验证（MFA）设备作为 IAM 用户登录时遇到问题，请与管理员联系以获取帮助。

作为管理员，您可以停用其他 IAM 用户的设备。此操作允许用户不使用 MFA 登录。如果 MFA 设备已更换或设备暂时不可用，您可以将此作为临时解决方案。但是，我们建议您尽快为用户启用新设备。要了解如何启用新 MFA 设备，请参阅 IAM 中的 Amazon 多重身份验证。

> **Note**
>
> 如果您使用 API 或 Amazon CLI 从 Amazon Web Services 账户 中删除用户，您必须停用或删除用户的 MFA 设备。您在删除用户的过程中执行此更改。有关删除用户的更多信息，请参阅移除或停用 IAM 用户。

主题

- 停用 MFA 设备（控制台）
- 停用 MFA 设备 (Amazon CLI)
- 停用 MFA 设备 (Amazon API)

停用 MFA 设备（控制台）

停用其他 IAM 用户的 MFA 设备（控制台）

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 要停用用户的 MFA 设备，请选择要删除其 MFA 的用户的名称。
4. 选择安全凭证选项卡。
5. 在多重身份验证（MFA）下，选择 MFA 设备旁边的单选按钮，选择删除，然后选择删除。

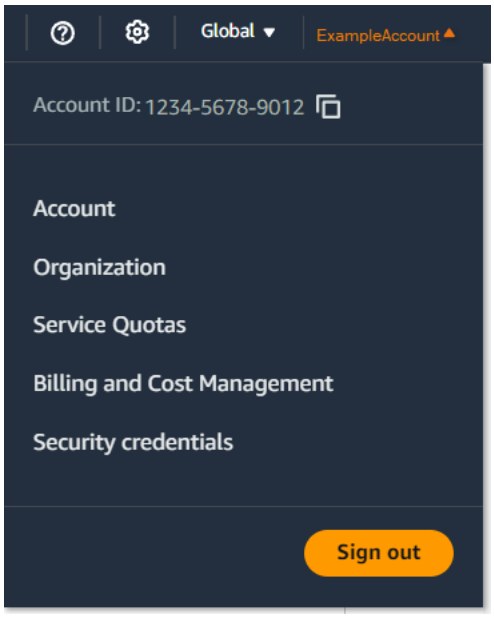
   随即从 Amazon 中删除该设备。该设备将不能用于登录或验证请求，直到它重新激活并与 Amazon 用户或 Amazon Web Services 账户根用户 相关联。

取消激活 Amazon Web Services 账户根用户的 MFA 设备（控制台）

1. 选择 Root user（根用户）并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录 IAM 控制台。在下一页上，输入您的密码。

> **Note**
>
> 作为根用户，您无法登录到以 IAM 用户身份登录页面。如果您看到以 IAM 用户身份登录页面，请选择该页面底部附近的使用根用户电子邮件登录。要获取以根用户身份登录方面的帮助，请参阅《Amazon 登录 用户指南》中的以根用户身份登录 Amazon Web Services 管理控制台。

2. 在导航栏的右侧选择您的账户名称，然后选择 Security credentials（安全凭证）。如有必要，选择 Continue to Security Credentials（继续使用安全凭证）。



3. 在多重身份验证 (MFA) 部分中，选择要停用的 MFA 设备旁边的单选按钮，然后选择 Remove（删除）。
4. 选择移除 。

   Amazon Web Services 账户 的 MFA 设备已停用。检查与您的 Amazon Web Services 账户 相关联的电子邮件邮箱中有无来自 Amazon Web Services 的确认邮件。该电子邮件向您通知 Amazon Web Services 多重身份验证 (MFA) 已停用。邮件将来自 `@amazon.com` 或 `@aws.amazon.com`。

> **Note**
>
> 当您通过 Amazon Web Services 管理控制台或在登录过程中添加新的虚拟 MFA 设备时，系统会删除您 Amazon Web Services 账户中未分配的虚拟 MFA 设备。未分配的虚拟 MFA 设备是

指您账户中的设备，但在登录过程中不被账户根用户或 IAM 用户使用。已删除这些设备，因此可以向您的账户添加新的虚拟 MFA 设备。它还允许您重复使用设备名称。

停用 MFA 设备 (Amazon CLI)

停用 IAM 用户的 MFA 设备 (Amazon CLI)

- 运行以下命令：`aws iam deactivate-mfa-device`

停用 MFA 设备 (Amazon API)

停用 IAM 用户的 MFA 设备 (Amazon API)

- 调用此操作：`DeactivateMFADevice`

在 IAM 中恢复受 MFA 保护的身份

若虚拟 MFA 设备或硬件 TOTP 令牌似乎运行正常，但您却无法使用它访问 Amazon 资源，则可能是与 Amazon 不同步所导致的。有关同步虚拟 MFA 设备或硬件 MFA 设备的信息，请参阅重新同步虚拟和硬件 MFA 设备。FIDO 安全密钥不同步。

如果 Amazon Web Services 账户根用户 的 MFA 设备丢失、损坏或无法工作，您可以恢复账户访问权限。IAM 用户必须与管理员联系，才能将装置停用。

Important

建议您激活多台 MFA 设备。注册多台 MFA 设备有助于确保在设备丢失或损坏时继续访问。您的 Amazon Web Services 账户根用户 和 IAM 用户最多可以注册 8 台任何类型的 MFA 设备。

先决条件 – 使用其他 MFA 设备

如果您的多重身份验证（MFA）设备丢失、损坏或无法正常工作，您可以使用已注册到同一根用户或 IAM 用户的另一台 MFA 设备登录。

要使用其他 MFA 设备登录

1. 使用您的 Amazon Web Services 账户 ID 或账户别名和密码登录 Amazon Web Services 管理控制台。

2. 在需要其他验证页面或多重身份验证页面上，选择尝试另一种 MFA 方法。
3. 使用您选择的 MFA 设备类型进行身份验证。
4. 根据您是否使用备用 MFA 设备成功登录，下一步会有所不同。
   - 如果您已成功登录，则可以 重新同步虚拟和硬件 MFA 设备，这可能会解决问题。如果 MFA 设备丢失或损坏，则您可以将其停用。有关停用任何 MFA 设备类型的说明，请参阅停用 MFA 设备。
   - 如果您无法使用 MFA 登录，则请使用 恢复根用户用户 MFA 设备 或 恢复 IAM 用户 MFA 设备 中的步骤恢复受 MFA 保护的身份。

恢复根用户用户 MFA 设备

如果您无法使用 MFA 登录，则可以使用其他身份验证方法登录，方法是使用在您的账户中注册的电子邮件和主要联系电话号码验证您的身份。

作为根用户使用替代的身份验证因素登录之前，确认您能够访问与您的账户关联的电子邮件和主要联系人电话号码。如果您需要更新主要联系人电话号码，则请以具有管理员访问权限的 IAM 用户身份而非根用户身份登录。有关更新账户联系信息的其他说明，请参阅《Amazon Billing 用户指南》中的编辑联系信息。如果您无权访问电子邮件和主要联系人电话号码，则必须联系 Amazon Web Services 支持。

> **Important**
>
> 我们建议您不断更新与根用户关联的电子邮件地址和联系电话号码，以便成功恢复账户。有关更多信息，请参阅《Amazon 账户管理 参考指南》中的 更新 Amazon Web Services 账户的主要联系人。

使用替代的身份验证因素作为 Amazon Web Services 账户根用户登录

1. 选择根用户并输入您的 Amazon Web Services 账户 电子邮件地址，以账户拥有者身份登录。Amazon Web Services 管理控制台在下一页上，输入您的密码。
2. 在需要其他验证页面上，选择要用于身份验证的 MFA 方法，然后选择下一步。

> **Note**
>
> 可能会显示替代文本，例如 Sign in using MFA（使用 MFA 登录）、Troubleshoot your authentication device（对身份验证设备进行故障排除）或 Troubleshoot MFA（对 MFA

进行故障排除），但其功能均相同。如果您无法使用替代身份验证因素验证您的账户电子邮件地址和主要联系人电话号码，则请与 Amazon Web Services 支持 联系以停用您的 MFA 设备。

3. 根据您使用的 MFA 类型，您将看到不同的页面，但是 MFA 问题排查选项的功能相同。在需要其他验证页面或多重身份验证页面上，选择 MFA 问题排查。
4. 如果需要，请再次键入您的密码，然后选择 Sign in。
5. 在身份验证设备问题排查页面的使用替代身份验证因素登录部分中，选择使用替代因素登录。
6. 使用替代身份验证因素登录页面上，通过验证电子邮件地址完成账户身份验证，然后选择发送验证电子邮件。
7. 检查与您的 Amazon Web Services 账户 关联的电子邮件中有无来自 Amazon Web Services（recover-mfa-no-reply@verify.signin.aws）的邮件。按照电子邮件中的指导进行操作。

   如果您没有在账户中看到该电子邮件，请检查垃圾邮件文件夹，或者返回到浏览器并选择 Resend the email。
8. 在验证您的电子邮件地址后，您可以继续验证您的账户的身份。要验证您的主要联系人电话号码，请选择 Call me now（立即呼叫我）。
9. 接听 Amazon 打来的电话，在听到提示时，在手机键盘上输入从 Amazon 网站获得的 6 位数号码。

   如果您没有接到 Amazon 打来的电话，请选择 Sign in (登录) 以再次登录到控制台并重新开始。或者，请参阅多重身份验证（MFA）设备丢失或无法使用联系支持人员以获取帮助。
10. 在验证您的电话号码后，您可以通过选择 Sign in to the console 登录到您的账户。
11. 下一步取决于您使用的 MFA 的类型：

    - 对于虚拟 MFA 设备，请从您的设备中删除账户。然后，转至 Amazon Security Credentials 页面并删除旧的 MFA 虚拟设备实体，然后再创建一个新的。
    - 对于 FIDO 安全密钥，请转至 Amazon Security Credentials（安全凭证）页面并停用旧 FIDO 密钥然后再启用一个新密钥。
    - 对于硬件 TOTP 令牌，请联系第三方提供商以帮助您修复或更换设备。您可以继续使用替代的身份验证因素登录，直到您收到新设备为止。在您拥有新的硬件 MFA 设备后，请转至 Amazon 安全凭证页面并删除旧的 MFA 设备。

> **Note**
>
> 不必将丢失或被盗的 MFA 设备替换为相同类型的设备。例如，如果 FIDO 安全密钥损坏，您订购了一个新的密钥，则可以使用虚拟 MFA 或硬件 TOTP 令牌，直到新 FIDO 安全密钥到达。

> **Important**
>
> 如果您的 MFA 设备丢失或被盗，则请在登录并建立替换 MFA 设备后更改您的根用户密码。攻击者可能窃取了身份验证设备，也可能拥有您当前的密码。有关更多信息，请参阅 更改 Amazon Web Services 账户根用户密码。

恢复 IAM 用户 MFA 设备

如果您是无法使用 MFA 登录的 IAM 用户，则无法自行恢复 MFA 设备。您必须与管理员联系，才能将设备停用。然后，您可以启用新设备。

作为 IAM 用户获取有关 MFA 设备的帮助

1. 请联系 Amazon 管理员或其他为您提供 IAM 用户的用户名和密码的相关人员。管理员必须停用 MFA 设备 (如停用 MFA 设备中所述)，以便您能够登录。
2. 下一步取决于您使用的 MFA 的类型：

   - 对于虚拟 MFA 设备，请从您的设备中删除账户。然后，启用虚拟设备，如在 Amazon Web Services 管理控制台 中分配虚拟 MFA 设备中所述。
   - 对于 FIDO 安全密钥，请联系第三方提供商帮助您更换设备。当您收到新的 FIDO 安全密钥时，请将其启用，如 在 Amazon Web Services 管理控制台 中分配通行密钥或安全密钥 中所述。
   - 对于硬件 TOTP 令牌，请联系第三方提供商以帮助您修复或更换设备。在您拥有新的物理 MFA 设备后，请启用设备，如在 Amazon Web Services 管理控制台 中分配硬件 TOTP 令牌中所述。

> **Note**
>
> 不必将丢失或被盗的 MFA 设备替换为相同类型的设备。您最多可以拥有 8 台任意组合的 MFA 设备。例如，如果 FIDO 安全密钥损坏，您订购了一个新的密钥，则可以使用虚拟 MFA 或硬件 TOTP 令牌，直到新 FIDO 安全密钥到达。

3. 如果您的 MFA 设备丢失或被盗，还请更改密码，以防攻击者盗走您的身份验证设备及还可能拥有您当前的密码。有关更多信息，请参阅 管理 IAM 用户的密码。

使用 MFA 保护 API 访问

利用 IAM policy，可以指定用户可调用哪些 API 操作。您可以通过要求用户使用多重身份验证（MFA）进行身份验证，然后才允许他们执行特别敏感的操作，从而应用额外的安全性。

例如，您可能拥有允许用户执行 Amazon EC2 `RunInstances`、`DescribeInstances` 和 `StopInstances` 操作的策略。但您可能希望限制破坏性操作（如 `TerminateInstances`），并确保用户只能在使用 Amazon MFA 设备进行身份验证后执行该操作。

主题

- 概述
- 方案：跨账户委派的 MFA 保护
- 方案：当前账户中 API 操作访问的 MFA 保护
- 方案：拥有基于资源的策略的资源的 MFA 保护

概述

向 API 操作添加 MFA 保护包括以下任务：

1. 管理员为每个用户配置 Amazon MFA 设备，这些用户必须发出要求 MFA 身份验证的 API 请求。有关更多信息，请参阅 IAM 中的 Amazon 多重身份验证。
2. 管理员为用户创建包含 `Condition` 元素的策略，以检查用户是否已使用 Amazon MFA 设备进行身份验证。
3. 用户会调用支持 MFA 参数的 Amazon STS API 操作之一：AssumeRole 或 GetSessionToken。调用中包含与用户关联的设备的设备标识符，以及该设备生成的基于时间的一次性密码 (TOTP)。在任一情况下，用户都会取回稍后用来向 Amazon 发出其他请求的临时安全凭证。

> **Note**
>
> 仅在服务支持临时安全凭证时，该服务的 API 操作的 MFA 保护才可用。有关这些服务的列表，请参阅使用临时安全凭证访问 Amazon。

如果授权失败，Amazon 将返回“访问被拒绝”错误消息（与任何未授权的访问一样）。由于采用受 MFA 保护的 API 策略，因此，如果用户在未获得有效的 MFA 身份验证的情况下试图调用 API 操作，则 Amazon 将拒绝用户访问策略中指定的 API 操作。如果 API 操作请求的时间戳在策略中指定的允许范围之外，也会拒绝操作。用户必须使用 MFA 代码和设备序列号请求新的临时安全凭证，通过 MFA 重新进行身份验证。

带 MFA 条件的 IAM policy

带 MFA 条件的策略可附加到以下项：

- 对于 IAM 用户或组：
- Amazon S3 存储桶、Amazon SQS 队列或 Amazon SNS 主题等资源
- 可由用户担任的 IAM 角色的信任策略

可使用策略中的 MFA 条件检查以下属性：

- Existence（存在性）- 如果只是验证用户是否已使用 MFA 进行身份验证，请检查 `aws:MultiFactorAuthPresent` 密钥在 `Bool` 条件中是否为 `True`。仅当用户使用短期凭证验证时，才会有该键。长期凭证，例如访问密钥，不包括此键。
- Duration (持续时间) - 如果您只希望在 MFA 身份验证后的指定时间内授予访问权限，请使用数值条件类型将 `aws:MultiFactorAuthAge` 密钥的有效期与某个值（如 3600 秒）进行比较。请注意，如果未使用 MFA，则 `aws:MultiFactorAuthAge` 键不会显示。

以下示例说明了 IAM 角色的信任策略，该策略包含一个 MFA 条件，用于测试是否存在 MFA 身份验证。利用此策略，`Principal` 元素中指定的 Amazon Web Services 账户 中的用户（将 `ACCOUNT-B-ID` 替换为有效的 Amazon Web Services 账户 ID）能够代入此策略附加的角色。但是，此类用户仅在已使用 MFA 进行身份验证的情况下才能够担任角色。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "ACCOUNT-B-ID"},
    "Action": "sts:AssumeRole",
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }
}
```

有关 MFA 的条件类型的更多信息，请参阅Amazon 全局条件上下文密钥、数字条件运算符和用于检查条件键是否存在的条件运算符。

在 GetSessionToken 和 AssumeRole 之间选择

Amazon STS 提供了两个 API 操作，可供用户传递 MFA 信息：GetSessionToken 和 AssumeRole。用户调用哪个 API 操作来获取临时安全凭证，取决于采用以下哪个方案。

将 **GetSessionToken** 用于以下方案：

- 调用访问 Amazon Web Services 账户 中的资源的 API 操作，该账户与发出请求的 IAM 用户相同。请注意，来自 GetSessionToken 请求的临时凭证仅在您将 MFA 信息包含在凭证请求中时才能访问 IAM 和 Amazon STS API 操作。由于 GetSessionToken 返回的临时凭证包含 MFA 信息，因此您可以检查由该凭证发出的单个 API 操作中的 MFA。
- 访问受基于资源且包含 MFA 条件的策略所保护的资源。

GetSessionToken 操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证操作。

将 **AssumeRole** 用于以下方案：

- 调用访问相同或不同 Amazon Web Services 账户 中的资源的 API 操作。API 调用可包含任何 IAM 或 Amazon STS API。请注意，要保护访问，您可以在用户担任角色时强制执行 MFA。由 AssumeRole 返回的临时凭证未将 MFA 信息包含在上下文中，因此您无法检查单个 API 操作的 MFA。这就是您必须使用 GetSessionToken 限制对受基于资源的策略保护的资源的访问的原因。

> **Note**
>
> IAM 用户使用 MFA 登录时，Amazon CloudTrail 日志将包含 MFA 信息。如果 IAM 用户代入某个 IAM 角色，CloudTrail 还将在利用代入角色执行操作的 `sessionContext` 属性中记录 `mfaAuthenticated: true`。但是，当使用代入角色的凭证进行 API 调用时，CloudTrail 日志记录与 IAM 要求的内容是分开的。有关更多信息，请参阅 CloudTrail userIdentity 元素。

本文档稍后将提供有关如何实现这些方案的详细信息。

有关受 MFA 保护的 API 访问的要点

了解 API 操作的 MFA 保护的以下几个方面非常重要：

- MFA 保护仅通过使用临时安全凭证提供，而该凭证必须使用 `AssumeRole` 或 `GetSessionToken` 获取。
- 无法将受 MFA 保护的 API 访问与 Amazon Web Services 账户根用户凭证配合使用。
- 无法将受 MFA 保护的 API 访问与 U2F 安全密钥配合使用。
- 将 MFA 设备与 Amazon 服务配合使用时，无法向联合身份用户分配这些设备；因此，这些用户无法访问受 MFA 控制的 Amazon 资源。(请参阅下一个要点。)
- 返回临时凭证的其他 Amazon STS API 操作不支持 MFA。对于 `AssumeRoleWithWebIdentity` 和 `AssumeRoleWithSAML`，用户通过外部提供程序进行身份验证，并且 Amazon 无法确定该提供程序是否需要 MFA。对于 `GetFederationToken`，MFA 不一定要与特定用户相关联。
- 同样，长期凭证（IAM 用户访问密钥和根用户访问密钥）无法用于受 MFA 保护的 API 访问，因为它们不会过期。
- 此外，可以在没有 MFA 信息的情况下调用 `AssumeRole` 和 `GetSessionToken`。在此情况下，发起人将取回临时安全凭证，但这些临时凭证的会话信息不指示使用 MFA 进行身份验证的用户。
- 要建立 API 操作的 MFA 保护，可将 MFA 条件添加到策略。策略必须包含 `aws:MultiFactorAuthPresent` 条件键以强制使用 MFA。对于跨账户委派，角色的信任策略必须包含条件键。
- 如果您允许其他 Amazon Web Services 账户 访问您账户中的资源，则您的资源安全性取决于受信任账户（另一个账户，而不是您的账户）的配置。即使在您强制实施多重身份验证时，也是如此。有权创建虚拟 MFA 设备的受信任账户中的任何标识都可以构建 MFA 索赔，以满足您角色的信任策略的该部分。在允许其他账户的成员访问必须进行多重身份验证的 Amazon 资源之前，您应确保受信任账户的所有者遵循安全最佳实践。例如，受信任账户应仅允许特定的受信任身份访问敏感 API 操作，例如 MFA 设备管理 API 操作。

- 如果策略包含 MFA 条件，则在以下情况下，将拒绝请求：用户未进行 MFA 身份验证或用户提供了无效的 MFA 设备标识符或无效的 TOTP。

方案：跨账户委派的 MFA 保护

在此方案中，您希望将访问权委托给另一账户中的 IAM 用户，但前提是该用户已使用 Amazon MFA 设备进行身份验证。有关跨账户委派的更多信息，请参阅 角色术语和概念。

假设您有一个账户 A（拥有待访问资源的信任账户）以及 IAM 用户 Anaya，她拥有管理员权限。她希望向账户 B（受信任账户）中的用户 Richard 授予访问权，但希望确保 Richard 在担任该角色之前已使用 MFA 进行身份验证。

1. 在信任账户 A 中，Anaya 创建一个名为 `CrossAccountRole` 的 IAM 角色，并将该角色的信任策略中的主体设置为账户 B 的账户 ID。此信任策略授予对 Amazon STS `AssumeRole` 操作的权限。Anaya 还向信任策略添加 MFA 条件，如以下示例中所示。

   JSON

   ```
   {
     "Version":"2012-10-17",
     "Statement": {
       "Effect": "Allow",
       "Principal": {"AWS": "ACCOUNT-B-ID"},
       "Action": "sts:AssumeRole",
       "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
     }
   }
   ```

2. Anaya 在该角色中添加一个权限策略，以指定允许该角色执行的操作。具有 MFA 保护的角色权限策略与任何其他角色权限策略没有差别。以下示例说明了 Anaya 添加到角色的策略；该策略允许担任该角色的用户对账户 A 中的表 `Books` 执行任何 Amazon DynamoDB 操作。此策略还允许 `dynamodb:ListTables` 操作，此操作是在控制台中执行操作所必需的。

   > **Note**
   >
   > 该权限策略不包含 MFA 条件。了解 MFA 身份验证仅用于确定用户是否可以担任此角色很重要。在用户担任此角色后，将不会进行进一步的 MFA 检查。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TableActions",
            "Effect": "Allow",
            "Action": "dynamodb:*",
            "Resource": "arn:aws:dynamodb:*:111122223333:table/Books"
        },
        {
            "Sid": "ListTables",
            "Effect": "Allow",
            "Action": "dynamodb:ListTables",
            "Resource": "*"
        }
    ]
}
```

3. 在受信任账户 B 中，管理员确保已使用 Amazon MFA 设备配置 IAM 用户 Richard，并且该用户知道设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
4. 在账户 B 中，管理员将以下策略附加到用户 Richard（或该用户所在的组），该策略允许该用户调用 `AssumeRole` 操作。资源被设置到 Anaya 在第 1 步中创建的角色的 ARN。注意，该策略不包含 MFA 条件。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sts:AssumeRole"
            ],
            "Resource": [
                "arn:aws:iam::111122223333:role/CrossAccountRole"
            ]
```

```
        }
    ]
}
```

5. 在账户 B 中，Richard（或 Richard 正在运行的应用程序）调用 AssumeRole。API 调用包含要担任角色的 ARN (arn:aws:iam::*ACCOUNT-A-ID*:role/CrossAccountRole)、MFA 设备的 ID 和 Richard 从其设备中获取的当前 TOTP。

   当 Richard 调用 AssumeRole 时，Amazon 将确定他是否拥有有效的凭证，包括 MFA 要求。如果是这样的话，Richard 将成功担任角色，并且可在使用角色的临时凭证时对账户 A 中名为 Books 的表执行任何 DynamoDB 操作。

   有关调用 AssumeRole 的程序的示例，请参阅使用 MFA 身份验证调用 AssumeRole。

方案：当前账户中 API 操作访问的 MFA 保护

在此方案中，您应确保仅当您 Amazon Web Services 账户 中的用户已使用 Amazon MFA 设备进行身份验证后才能访问敏感 API 操作。

假设您拥有账户 A，其中包含一组需要使用 EC2 实例的开发人员。普通开发人员可以使用实例，但他们未获得 ec2:StopInstances 或 ec2:TerminateInstances 操作的权限。您希望仅允许几个受信任用户执行这些“破坏性”特权操作，因此您将 MFA 保护添加到允许这些敏感 Amazon EC2 操作的策略中。

在此方案中，用户 Sofía 是受信任用户之一。用户 Anaya 是账户 A 中的管理员。

1. Anaya 确保已使用 Amazon MFA 设备配置 Sofía，并且 Sofía 知道该设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
2. Anaya 创建一个名为 EC2-Admins 的组并将用户 Sofía 添加到该组中。
3. Anaya 将以下策略附加到 EC2-Admins 组。此策略授予用户调用 Amazon EC2 StopInstances 和 TerminateInstances 操作的权限，但前提是该用户已使用 MFA 进行身份验证。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": [
      "ec2:StopInstances",
      "ec2:TerminateInstances"
```

```
    ],
    "Resource": ["*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

4.

> **Note**
>
> 要使此策略生效，用户必须先注销，然后重新登录。

如果用户 Sofía 需要停止或终止 Amazon EC2 实例，则她（或她正在运行的应用程序）可调用 `GetSessionToken`。此 API 操作传递 MFA 设备的 ID 和 Sofía 从其设备获取的当前 TOTP。

5. 用户 Sofía（或 Sofía 正在使用的应用程序）使用由 `GetSessionToken` 提供的临时凭证来调用 Amazon EC2 `StopInstances` 或 `TerminateInstances` 操作。

   有关调用 `GetSessionToken` 的程序的示例，请参阅本文档后面的 使用 MFA 身份验证调用 GetSessionToken。

方案：拥有基于资源的策略的资源的 MFA 保护

在此方案中，您是 S3 存储桶、SQS 队列或 SNS 主题的所有者。您希望确保访问资源的任何 Amazon Web Services 账户 中的任何用户都已使用 Amazon MFA 设备进行身份验证。

此方案介绍了一种提供跨账户 MFA 保护的方法，无需用户先担任角色。在此情况下，用户可在满足以下三个条件时访问资源：用户已使用 MFA 进行身份验证、能够从 `GetSessionToken` 获取临时安全凭证且在受资源策略信任的账户中。

假设您在账户 A 中并创建一个 S3 存储桶。您希望向几个不同 Amazon Web Services 账户 中的用户授予对此存储桶的访问权，但前提是这些用户已使用 MFA 进行身份验证。

在此方案中，用户 Anaya 是账户 A 中的管理员。用户 Nikhil 是账户 C 中的 IAM 用户。

1. 在账户 A 中，Anaya 创建一个名为 `Account-A-bucket` 的存储桶。
2. Anaya 向该存储桶添加存储桶策略。该策略允许账户 A、账户 B 或账户 C 中的所有用户执行存储桶中的 Amazon S3 `PutObject` 和 `DeleteObject` 操作。该策略包含 MFA 条件。

   JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": [
      "ACCOUNT-A-ID",
      "ACCOUNT-B-ID",
      "ACCOUNT-C-ID"
    ]},
    "Action": [
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": ["arn:aws:s3:::ACCOUNT-A-BUCKET-NAME/*"],
    "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
  }]
}
```

> **Note**
>
> Amazon S3（仅）针对根账户访问提供“MFA 删除”功能。在您设置存储桶的版本化状态时，可启用 Amazon S3 MFA Delete 功能。Amazon S3 MFA Delete 功能不适用于 IAM 用户，在管理时独立于 MFA 保护的 API 访问。即使 IAM 用户具有删除存储桶的权限，但在启用 Amazon S3 MFA Delete 功能时，也无法执行删除。有关 Amazon S3 MFA Delete 功能的更多信息，请参阅 MFA Delete。

3. 在账户 C 中，管理员确保已使用 Amazon MFA 设备配置用户 Nikhil，并且该用户知道设备的 ID。设备 ID 是序列号（如果是硬件 MFA 设备）或设备的 ARN（如果是虚拟 MFA 设备）。
4. 在账户 C 中，Nikhil（或他正在运行的应用程序）将调用 `GetSessionToken`。此调用包括 MFA 设备的 ID 或 ARN 以及 Nikhil 从其设备中获取的当前 TOTP。
5. Nikhil（或他正在使用的应用程序）使用 `GetSessionToken` 返回的临时凭证调用 Amazon S3 `PutObject` 操作以将文件上传到 `Account-A-bucket`。

   有关调用 `GetSessionToken` 的程序的示例，请参阅本文档后面的 使用 MFA 身份验证调用 GetSessionToken。

> **Note**
>
> 在此情况下，AssumeRole 返回的临时凭证将不可用。尽管用户可以提供 MFA 信息来担任角色，但 AssumeRole 返回的临时凭证不包含 MFA 信息。需要此信息才能满足策略中的 MFA 条件。

示例代码：使用多重验证请求凭证

以下示例说明如何调用 GetSessionToken 和 AssumeRole 操作并传递 MFA 身份验证参数。无需任何权限即可调用 GetSessionToken，但您必须拥有允许您调用 AssumeRole 的策略。随后，返回的凭证将用于列出账户中的所有 S3 存储桶。

使用 MFA 身份验证调用 GetSessionToken

以下示例说明了如何调用 GetSessionToken 并传递 MFA 身份验证信息。然后使用 GetSessionToken 操作返回的临时安全凭证来列出账户中的所有 S3 存储桶。

附加到运行此代码的用户（或用户所在的组）的策略提供了返回的临时凭证的权限。对于此示例代码，该策略必须向用户授予请求 Amazon S3 ListBuckets 操作的权限。

以下代码示例演示如何使用 GetSessionToken。

CLI

Amazon CLI

要获取 IAM 身份的一组短期凭证

以下 get-session-token 命令将检索进行调用的 IAM 身份的一组短期凭证。生成的凭证可用于策略要求多重身份验证（MFA）的请求。凭证在生成 15 分钟后过期。

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

输出：

```
{
```

```
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的请求临时安全凭证。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetSessionToken。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回包含在设定时间段内有效的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于请求临时凭证的凭证是根据当前 shell 默认值推断出来的。要指定其他凭证，请使用 -ProfileName 或 -AccessKey/-SecretKey 参数。

```
Get-STSSessionToken
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

示例 2：返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于发出请求的凭证是从指定的配置文件中获得的。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

示例 3：使用与其凭证在配置文件“myprofilename”中指定的账户关联的 MFA 设备的标识号和该设备提供的值，返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetSessionToken。

Tools for PowerShell V5

示例 1：返回包含在设定时间段内有效的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于请求临时凭证的凭证是根据当前 shell 默认值推断出来的。要指定其他凭证，请使用 -ProfileName 或 -AccessKey/-SecretKey 参数。

```
Get-STSSessionToken
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
```

```
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

示例 2：返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于发出请求的凭证是从指定的配置文件中获得的。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

输出：

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

示例 3：使用与其凭证在配置文件“myprofilename”中指定的账户关联的 MFA 设备的标识号和该设备提供的值，返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

输出：

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetSessionToken。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

通过传递 MFA 令牌获取会话令牌，然后使用令牌列出账户的 Amazon S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
```

```
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的 GetSessionToken。

使用 MFA 身份验证调用 AssumeRole

下面的示例说明了如何调用 AssumeRole 并传递 MFA 身份验证信息。然后使用 AssumeRole 返回的临时安全证书列出账户中的所有 Amazon S3 存储桶。

有关此方案的更多信息，请参阅方案：跨账户委派的 MFA 保护。

以下代码示例演示如何使用 AssumeRole。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
```

```
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do
 not specify a region,
        // or to use a specific region.
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);
```

```
            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 AssumeRole。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
#############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
#############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

#############################################################################
# function errecho
```

```
#
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
#############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
```

```
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 AssumeRole。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
```

```
                            const Aws::String &roleSessionName,
                            const Aws::String &externalId,
                            Aws::Auth::AWSCredentials &credentials,
                            const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 AssumeRole。

CLI

Amazon CLI

要代入角色

以下 assume-role 命令将检索 IAM 角色 s3-access-example 的一组短期凭证。

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

输出：

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
}
```

该命令的输出包含访问密钥、私有密钥和会话令牌，您可以使用它们对 Amazon 进行身份验证。

要使用 Amazon CLI，则可以设置与角色关联的命名配置文件。使用配置文件时，Amazon CLI 将调用 assume-role 并为您管理凭证。有关更多信息，请参阅《Amazon CLI 用户指南》中的在 Amazon CLI 中使用 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AssumeRole。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
```

```
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();

        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
 String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
```

```
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
 exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 AssumeRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

创建客户端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

代入 IAM 角色。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AssumeRole。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);
```

```
  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AssumeRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回一组临时凭证（访问密钥、秘密密钥和会话令牌），这些凭证可用于在一小时内访问请求用户通常可能无法访问的 Amazon 资源。返回的凭证具有所担任角色的访问策略和所提供策略允许的权限（您不能使用所提供策略授予超出所担任角色访问策略定义权限的权限）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

示例 2：返回一组有效期为一小时的临时凭证，其拥有的权限与所担任角色访问策略中定义的权限相同。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

示例 3：返回一组临时凭证，提供与用于执行 cmdlet 的用户凭证关联的 MFA 的序列号和生成的令牌。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

示例 4：返回一组临时凭证，其已承担客户账户中定义的角色。对于第三方可以担任的每个角色，客户账户必须使用每次担任该角色时都必须在 -ExternalId 参数中传递的标识符来创建角色。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AssumeRole。

Tools for PowerShell V5

示例 1：返回一组临时凭证（访问密钥、秘密密钥和会话令牌），这些凭证可用于在一小时内访问请求用户通常可能无法访问的 Amazon 资源。返回的凭证具有所担任角色的访问策略和所提供策略允许的权限（您不能使用所提供策略授予超出所担任角色访问策略定义权限的权限）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

示例 2：返回一组有效期为一小时的临时凭证，其拥有的权限与所担任角色访问策略中定义的权限相同。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

示例 3：返回一组临时凭证，提供与用于执行 cmdlet 的用户凭证关联的 MFA 的序列号和生成的令牌。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

示例 4：返回一组临时凭证，其已承担客户账户中定义的角色。对于第三方可以担任的每个角色，客户账户必须使用每次担任该角色时都必须在 -ExternalId 参数中传递的标识符来创建角色。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 AssumeRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

代入需要 MFA 令牌的 IAM 角色并使用临时凭证列出该账户的 Amazon S3 存储桶。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
```

```
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 AssumeRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
```

```
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 AssumeRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
```

```
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 AssumeRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSSTS

    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
```

```
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 AssumeRole。

IAM 用户的服务特定凭证

服务特定凭证是为特定 Amazon 服务设计的专用身份验证机制。与标准 Amazon 凭证相比，这些凭证提供简化的身份验证，并且是根据个别 Amazon 服务的身份验证要求量身定制的。与可以跨多个 Amazon 服务使用的访问密钥不同，服务特定凭证仅设计用于为其创建的服务。此针对性的方法通过限制凭证的范围来增强安全性。

服务特定凭证通常由用户名和密码对或根据特定服务的要求格式化的专用 API 密钥组成。创建服务特定凭证时，这些凭证默认处于活动状态，可以立即使用。针对每个 IAM 用户的每个受支持服务，最多可拥有两组服务特定凭证。此限制允许您在需要时轮换到新集合时保留一个活动集。Amazon 当前支持以下服务的服务特定凭证：

何时使用服务特定凭证

服务专用凭证旨在与第三方库、SDK、工具或应用程序兼容，这些库、SDK、工具或应用程序与 Amazon 凭证、Amazon SDK 或 Amazon API 本身不兼容。此类用例包括从自托管基础设施或其他提供商托管的服务迁移到 Amazon 服务。

从零开始时，只要有可能，我们建议您使用 Amazon 临时凭证（例如 IAM 角色提供的凭证），使用支持 Amazon 临时凭证的 Amazon SDK 或库对 Amazon 服务进行身份验证。

轮换服务特定凭证

作为最佳安全实践，请定期轮换服务特定凭证。要在不中断应用程序的情况下轮换凭证，请执行下面的操作：

1. 为同一服务和 IAM 用户创建第二组服务特定凭证

2. 更新所有应用程序以使用新凭证并验证它们是否正常工作
3. 将原始凭证的状态更改为“非活跃”
4. 验证所有应用程序是否仍正常运行
5. 当确信不再需要非活跃服务特定凭证时，请删除它们

监控服务特定凭证

您可以使用 Amazon CloudTrail 来监控 Amazon 账户中服务特定凭证的使用情况。要查看与服务特定凭证使用情况相关的 CloudTrail 事件，请查看 CloudTrail 日志中有关使用凭证的服务的事件。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。

为了提高安全性，请考虑设置 CloudWatch 警报，以便通知您可能表明未经授权访问或其他安全问题的特定凭证使用模式。有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的 Monitoring CloudTrail Log Files with Amazon CloudWatch Logs。

以下主题提供有关服务特定凭证的信息。

主题

- Amazon 服务的 API 密钥
- 将 IAM 与 Amazon Keyspaces（Apache Cassandra 兼容）结合使用

Amazon 服务的 API 密钥

一些 Amazon 服务除了支持标准 IAM 凭证（如临时安全凭证和长期访问密钥），也支持使用 API 密钥以对编程请求进行身份验证。 Amazon 提供两种类型的 API 密钥：

- 长期 API 密钥 – 长期 API 密钥与 IAM 用户关联并使用 IAM 服务特定凭证生成。这些凭证仅设计用于单一 Amazon 服务，通过限制凭证范围来增强安全性。您可以设置长期 API 密钥的过期时间。要生成长期 API 密钥，可使用 IAM 或特定服务的控制台：Amazon CLI、或 Amazon API。
- 短期 API 密钥 – 短期 API 密钥是使用 Amazon 签名版本 4 的预签名 URL。短期 API 密钥与生成 API 密钥的身份的凭证具有相同的权限和过期时间，有效期最长为 12 小时或控制台会话的剩余时间（以较短者为准）。您可以使用 Amazon Bedrock/Amazon Web Services 上的 Claude Platform 控制台、Python 和其他编程语言的包来生成短期 API 密钥。有关更多信息，请参阅《Amazon Bedrock 用户指南》中的 Generate Amazon Bedrock API keys for easy access to the Amazon Bedrock API（生成 Amazon Bedrock API 密钥以便轻松访问 Amazon Bedrock API），以及《用户指南》中的身份验证。

> **Note**
>
> 与短期 API 密钥相比，长期 API 密钥的安全风险更高。建议尽可能使用短期 API 密钥或临时安全凭证。如果使用长期 API 密钥，建议实施定期密钥轮换实践。

支持 API 密钥的服务

下表列出了支持 API 密钥的 Amazon 服务以及每项服务支持的 API 密钥类型。

#	服务	长期 API 密钥	短期 API 密钥	托管策略自动附加	特定服务的文档
1	Amazon Bedrock	支持	是	AmazonBedrockLimitedAccess	使用 Amazon Bedrock API 密钥
2	Amazon Web Services 上的 Claude Platform	是	是	AnthropicInferenceAccess	身份验证
3	Amazon CloudWatch	是	不适用	CloudWatchAPIKeyAccess	为指标设置持有者令牌身份验证
4	Amazon CloudWatch Logs	是	不适用	CloudWatchLogsAPIKeyAccess	设置持有者令牌身份验证

当您为某项服务生成长期 API 密钥时，相应的 Amazon 托管策略会自动附加到 IAM 用户，从而授予访问该服务的核心操作权限。如果需要额外的访问权限，则可以修改 IAM 用户的权限。有关修改权限的信息，请参阅the section called “添加或删除身份权限”。

要了解有关特定服务的 API 密钥的更多信息，请参阅上表中特定服务的文档链接。

长期 API 密钥的先决条件

必须先满足以下先决条件，然后才能在 IAM 控制台生成长期 API 密钥：

- 与长期 API 密钥关联的 IAM 用户。有关创建 IAM 用户的说明，请参阅the section called “创建用户”。
- 您必须具备以下 IAM 策略权限来管理 IAM 用户的服务特定凭证。示例策略授予创建、列出、更新、删除和重置服务特定凭证的权限。请将 Resource 元素中的 *username* 值替换为要为其生成长期 API 密钥的 IAM 用户的名称：

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageBedrockServiceSpecificCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:UpdateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ResetServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/username"
        }
    ]
}
```

生成长期 API 密钥（控制台）

在 IAM 控制台中为特定服务生成长期 API 密钥

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择用户。
3. 选择要为其生成长期 API 密钥的 IAM 用户。
4. 选择安全凭证选项卡。
5. 在 API 密钥部分中，选择生成 API 密钥。
6. 从Amazon服务下拉列表中，选择您希望 API 密钥用于身份验证的服务。
7. 对于 API 密钥过期时间，请执行以下操作之一：

- 选择 API 密钥的过期时间 1、5、30、90 或 365 天。
- 选择自定义持续时间以指定自定义 API 密钥到期日期。
- 选择永不过期（不建议）。

8. 选择生成 API 密钥。
9. 复制或下载 API 密钥。这是唯一可以查看 API 密钥值的时间。

> **Important**
>
> 安全存储 API 密钥。关闭对话框后，将无法再次检索 API 密钥。如果丢失或忘记 API 密钥，则无法进行检索。相反，请生成新的 API 密钥并停用旧密钥。

生成长期 API 密钥（Amazon CLI）

要使用 Amazon CLI 生成长期 API 密钥，请按照以下步骤进行操作：

1. 使用 create-user 命令创建将与服务一起使用的 IAM 用户：

```
aws iam create-user \
    --user-name APIKeyUser_1
```

2. 使用 attach-user-policy 命令将 Amazon 托管式策略附加到 IAM 用户。

   对于 Amazon Bedrock：

```
aws iam attach-user-policy --user-name APIKeyUser_1 \
    --policy-arn arn:aws:iam::aws:policy/AmazonBedrockLimitedAccess
```

   对于 Amazon Web Services 上的 Claude Platform：

```
aws iam attach-user-policy --user-name APIKeyUser_1 \
    --policy-arn arn:aws:iam::aws:policy/AnthropicInferenceAccess
```

   对于 Amazon CloudWatch：

```
aws iam attach-user-policy --user-name APIKeyUser_1 \
    --policy-arn arn:aws:iam::aws:policy/CloudWatchAPIKeyAccess
```

对于 Amazon CloudWatch Logs：

```
aws iam attach-user-policy --user-name APIKeyUser_1 \
    --policy-arn arn:aws:iam::aws:policy/CloudWatchLogsAPIKeyAccess
```

3. 使用 create-service-specific-credential 命令生成长期 API 密钥。

对于 Amazon Bedrock：

```
aws iam create-service-specific-credential \
    --user-name APIKeyUser_1 \
    --service-name bedrock.amazonaws.com \
    --credential-age-days 30
```

对于 Amazon Web Services 上的 Claude Platform：

```
aws iam create-service-specific-credential \
    --user-name APIKeyUser_1 \
    --service-name aws-external-anthropic.amazonaws.com \
    --credential-age-days 30
```

对于 Amazon CloudWatch：

```
aws iam create-service-specific-credential \
    --user-name APIKeyUser_1 \
    --service-name cloudwatch.amazonaws.com \
    --credential-age-days 30
```

对于 Amazon CloudWatch Logs：

```
aws iam create-service-specific-credential \
    --user-name APIKeyUser_1 \
    --service-name logs.amazonaws.com \
    --credential-age-days 30
```

> **Note**
>
> `--credential-age-days` 参数是可选的。您可以指定 1 至 36600 天之间的值。如果省略此参数，API 密钥将不会过期。

响应中返回的 `ServiceApiKeyValue` 即为该服务的长期 API 密钥。安全地存储 `ServiceApiKeyValue` 值，因为您以后无法检索它。

列出长期 API 密钥 (Amazon CLI)

要列出特定用户的长期 API 密钥元数据，请使用带有 `--user-name` 参数的 list-service-specific-credentials 命令：

```
aws iam list-service-specific-credentials \
    --service-name bedrock.amazonaws.com \
    --user-name APIKeyUser_1
```

> **Note**
>
> 将 `bedrock.amazonaws.com` 替换为相应的服务名称（例如，Amazon CloudWatch Logs 替换为 `logs.amazonaws.com`，或者 Amazon Web Services 上的 Claude Platform 替换为 `aws-external-anthropic.amazonaws.com`）。

要列出账户中的所有长期 API 密钥元数据，请使用带有 `--all-users` 参数的 list-service-specific-credentials 命令：

```
aws iam list-service-specific-credentials \
    --service-name bedrock.amazonaws.com \
    --all-users
```

更新长期 API 密钥状态 (Amazon CLI)

要更新长期 API 密钥的状态，请使用 update-service-specific-credential 命令：

```
aws iam update-service-specific-credential \
    --user-name "APIKeyUser_1" \
    --service-specific-credential-id "ACCA1234EXAMPLE1234" \
```

```
    --status Inactive|Active
```

生成长期 API 密钥（Amazon API）

您可以使用以下 IAM API 操作来管理任何受支持服务的长期 API 密钥：

- CreateServiceSpecificCredential
- ListServiceSpecificCredentials
- UpdateServiceSpecificCredential
- DeleteServiceSpecificCredential
- ResetServiceSpecificCredential

短期 API 密钥（特定服务）

短期 API 密钥当前仅由特定服务支持。

有关通过 Amazon Bedrock 生成和使用短期 API 密钥的信息，请参阅《Amazon Bedrock 用户指南》中的生成 API 密钥。

有关 Amazon Web Services 上的 Claude Platform 的生成和使用短期 API 密钥的信息，请参阅《Amazon Web Services 上的 Claude Platform 用户指南》中的身份验证。

特定服务的信息

- 有关通过 Amazon Bedrock 使用 API 密钥的更多信息，请参阅《Amazon Bedrock 用户指南》中的使用 Amazon Bedrock API 密钥。
- 有关通过 Amazon Web Services 上的 Claude Platform 使用 API 密钥的更多信息，请参阅《Amazon Web Services 上的 Claude Platform 用户指南》中的身份验证。
- 有关在 Amazon CloudWatch 中使用 API 密钥的更多信息，请参阅《Amazon CloudWatch 用户指南》为指标设置持有者令牌身份验证。
- 有关通过 Amazon CloudWatch Logs 使用 API 密钥的更多信息，请参阅《Amazon CloudWatch Logs 用户指南》中的为指标设置持有者令牌身份验证。

将 IAM 与 Amazon Keyspaces（Apache Cassandra 兼容）结合使用

Amazon Keyspaces（用于 Apache Cassandra）是一种可扩展、高可用、托管的 Apache Cassandra 兼容数据库服务。您可以通过 Amazon Web Services 管理控制台 或以编程方式访问 Amazon Keyspaces。要使用服务特定的凭证以编程方式访问 Amazon Keyspaces，您可以使用 `cqlsh` 或开源

Cassandra 驱动程序。服务特定的凭证包括用户名和密码，例如 Cassandra 用于身份验证和访问管理的用户名和密码。针对每个用户的每个受支持服务，您最多可拥有两组服务特定凭证。

要使用 Amazon 访问密钥以编程方式访问 Amazon Keyspaces，您可以使用 Amazon 开发工具包、Amazon Command Line Interface（Amazon CLI）或带有 SiGv4 插件的开源 Cassandra 驱动程序。要了解更多信息，请参阅《Amazon Keyspaces (for Apache Cassandra) Developer Guide》中的 Create and configure Amazon credentials for Amazon Keyspaces。

> **Note**
>
> 如果您计划仅通过控制台与 Amazon Keyspaces 交互，则无需生成服务特定凭证。有关更多信息，请参阅《Amazon Keyspaces（Apache Cassandra 兼容）开发人员指南》中的使用控制台访问 Amazon Keyspaces 。

有关访问 Amazon Keyspaces 所需权限的更多信息，请参阅 Amazon Keyspaces (for Apache Cassandra) 开发人员指南中的 Amazon Keyspaces (for Apache Cassra) 基于身份的策略示例。

生成 Amazon Keyspaces 凭证（控制台）

您可以使用 Amazon Web Services 管理控制台 为 IAM 用户生成 Amazon Keyspaces 用户（针对 Apache Cassandra）凭证。

生成 Amazon Keyspaces 服务特定凭证（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Users (用户)，然后选择需要凭证的用户的名称。
3. 在 Credentials for Amazon Keyspaces (for Apache Cassandra)（Amazon Keyspaces (for Apache Cassandra) 凭证）下方的 Security Credentials（安全凭证）选项卡中，选择 Generate credentials（生成凭证）。
4. 您的服务特定凭证现在可用。这是唯一一次查看或下载密码的机会。以后您无法恢复它。不过，您可以随时重置密码。将用户和密码保存在安全的位置，因为随后您需要使用它们。

生成 Amazon Keyspaces 凭证 (Amazon CLI)

您可以使用 Amazon CLI 为 IAM 用户生成 Amazon Keyspaces 用户（针对 Apache Cassandra）凭证。

生成 Amazon Keyspaces 服务特定凭证 (Amazon CLI)

- 使用以下命令：
  - aws iam create-service-specific-credential

生成 Amazon Keyspaces 凭证 (Amazon API)

您可以使用 Amazon API 为 IAM 用户生成 Amazon Keyspaces 用户（针对 Apache Cassandra）凭证。

生成 Amazon Keyspaces 服务特定凭证 (Amazon API)

- 完成以下操作：
  - CreateServiceSpecificCredential

查找未使用的 Amazon 凭证

为提高您的 Amazon Web Services 账户 的安全性，请删除不需要的 IAM 用户凭证（即密码和访问密钥）。例如，当用户离开您的组织或不再需要访问 Amazon 时，请找到用户所使用的凭证并确保这些凭证不再有效。理想情况下，如果不再需要凭证，可将其删除。您始终可在将来需要这些凭证时创建它们。您至少应该更改密码或停用访问密钥，使以前的用户不再具有访问权限。

当然，未使用 这个词可以有多种理解，但通常指在指定时段内没用过的凭证。

查找未使用的密码

您可以使用 Amazon Web Services 管理控制台查看用户的密码使用信息。如果您有大量的用户，则您可以使用控制台下载包含有关每个用户上次使用其控制台密码的时间信息的凭证报告。您还可以从 Amazon CLI 或 IAM API 访问信息。

查找未使用的密码 (控制台)

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 如有必要，请将 Console last sign-in 列添加到用户表中：

a. 在最右侧的表上方，选择设置图标
( )。

b. 在选择可见列中，选择控制台上次登录。

c. 选择确认，返回到用户列表。

4. Console last sign-in（控制台上次登录时间）列显示用户上次通过控制台登录 Amazon 的日期。您可以利用该信息查找超过指定时间段未使用密码登录的用户。对于从未使用密码登录的用户，该列显示 Never。None 表示没有密码的用户。最近未使用的密码可能适合做删除处理。

> **Important**
>
> 由于服务问题，上次使用密码的数据不包括从 2018 年 5 月 3 日 22:50（太平洋时间）到 2018 年 5 月 23 日 14:08（太平洋时间）之间的密码使用。这会影响 IAM 控制台中显示的上次登录日期和 IAM 凭证报告中以及 GetUser API 操作返回的上次使用密码的日期。如果用户在受影响的时间内登录，则返回的上次使用密码的日期是用户在 2018 年 5 月 3 日之前最后一次登录的日期。对于在 2018 年 5 月 23 日 14:08（太平洋时间）之后登录的用户，所返回的上次使用密码的日期准确无误。
> 如果您使用上次使用密码的信息来识别未使用的凭证以将其删除（如删除在过去 90 天内未登录过 Amazon 的用户），建议您调整评估时段以纳入 2018 年 5 月 23 日之后的日期。或者，如果您的用户使用访问密钥以编程方式访问 Amazon，则您可以参阅上次使用访问密钥的信息，因为其中的所有日期都准确无误。

通过下载凭证报告查找未使用的密码 (控制台)

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择凭证报告。
3. 选择 Download Report 以下载名为 `status_reports_<date>T<time>.csv` 的逗号分隔值 (CSV) 文件。第五列包含具有日期或下列项之一的 `password_last_used` 列：

   - N/A - 根本未获得密码的用户。
   - no_information - 自 在 2014 年 10 月 20 日 IAM 开始跟踪密码使用期限以来未使用其密码的用户。

查找未使用的密码 (Amazon CLI)

运行以下命令查找未使用的密码：

- `aws iam list-users` 返回用户列表，每个用户均有一个 `PasswordLastUsed` 值。如果缺少此值，则用户未获得密码或自 IAM 在 2014 年 10 月 20 日开始跟踪密码使用期限以来未使用密码。

查找未使用的密码 (Amazon API)

调用以下操作查找未使用的密码：

- `ListUsers` 返回用户集合，每个用户均有一个 `<PasswordLastUsed>` 值。如果缺少此值，则用户未获得密码或自 IAM 在 2014 年 10 月 20 日开始跟踪密码使用期限以来未使用密码。

有关用于下载凭证报告的命令的信息，请参阅获取凭证报告 (Amazon CLI)。

查找未使用的访问密钥

您可以使用 Amazon Web Services 管理控制台查看用户的访问密钥使用信息。如果您有大量的用户，则您可以使用控制台下载凭证报告，以查找每个用户上次使用其访问密钥的时间。您还可以从 Amazon CLI 或 IAM API 访问信息。

查找未使用的访问密钥 (控制台)

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 如有必要，将 Access key last used 列添加到用户表中：
   a. 在最右侧的表上方，选择设置图标 ( )。
   b. 在选择可见列中，选择上次使用的访问密钥。
   c. 选择确认，返回到用户列表。
4. Access key last used (访问密钥上次使用) 列显示用户上次以编程方式访问 Amazon 以来经过的天数。您可以利用该信息查找超过指定时间段未使用访问密钥的用户。对于没有访问密钥的用户，该列显示 –。最近未使用的访问密钥可能适合做删除处理。

通过下载凭证报告来查找未使用的访问密钥 (控制台)

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择凭证报告。
3. 选择 Download Report 以下载名为 `status_reports_<date>T<time>.csv` 的逗号分隔值 (CSV) 文件。列 11 至 13 包含访问密钥 1 的上次使用日期、区域和服务信息。列 16 至列 18 包含访问密钥 2 的相同信息。如果用户没有访问密钥或用户自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期限以来未使用访问密钥，则值为 N/A。

查找未使用的访问密钥 (Amazon CLI)

运行以下命令查找未使用的访问密钥：

- `aws iam list-access-keys` 返回有关用户的访问密钥的信息，包括 `AccessKeyID`。
- `aws iam get-access-key-last-used` 采用访问密钥 ID 并返回输出，包括 `LastUsedDate`、`Region` (上次在其中使用访问密钥) 和上一次请求的服务的 `ServiceName`。如果缺少 `LastUsedDate`，则自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期限以来未使用访问密钥。

查找未使用的访问密钥 (Amazon API)

调用以下操作查找未使用的访问密钥：

- `ListAccessKeys` 返回与指定用户关联的访问密钥的 `AccessKeyID` 值的列表。
- `GetAccessKeyLastUsed` 采用访问密钥 ID 并返回值的集合。包括 `LastUsedDate`、`Region` (上次在其中使用访问密钥) 和上次请求的服务的 `ServiceName`。如果缺少此值，则用户未获得访问密钥或自 IAM 在 2015 年 4 月 22 日开始跟踪访问密钥使用期间以来未使用访问密钥。

有关用于下载凭证报告的命令的信息，请参阅获取凭证报告 (Amazon CLI)。

为您的 Amazon Web Services 账户 生成凭证报告

您可以生成和下载列出您账户中所有用户及其各个凭证状态（包括密码、访问密钥和 MFA 设备）的凭证报告。您可以从 Amazon Web Services 管理控制台、Amazon 开发工具包和命令行工具或 IAM API 获取凭证报告。

> **Note**
>
> IAM 凭证报告仅包含以下 IAM 管理的凭证：密码、每个用户的前两个访问密钥、MFA 设备和 X.509 签名证书。该报告不包括服务特定的凭证（例如 CodeCommit 密码、Amazon Bedrock 长期 API 密钥或 Amazon CloudWatch Logs 长期 API 密钥）或除前两个以外的任何其他用户访问密钥。要查看完整的凭证，请使用 ListServiceSpecificCredentials 和 ListAccessKeys API。

您可以使用凭证报告帮助您进行审核和合规性工作。您可以使用该报告来审计凭证生命周期要求的效果，如密码和访问密钥更新要求。您可以向外部审核员提供报告，或向审核员授予权限，以便他或她可以直接下载报告。

您可以按每四小时一次的频率生成凭证报告。请求报告时，IAM 先检查在过去四小时内是否生成了 Amazon Web Services 账户 的报告。如果是，则下载最新的报告。如果账户的最新报告是四小时前生成的，或者账户以前没有报告，则 IAM 将生成并下载新报告。

主题

- 所需的权限
- 了解报告格式
- 获取凭证报告（控制台）
- 获取凭证报告 (Amazon CLI)
- 获取凭证报告 (Amazon API)

所需的权限

创建和下载报告需要以下权限：

- 创建凭证报告：`iam:GenerateCredentialReport`
- 下载报告：`iam:GetCredentialReport`

了解报告格式

凭证报告采用逗号分隔值 (CSV) 文件格式。您可以使用常用电子表格软件打开 CSV 文件以执行分析，也可以构建应用程序以编程方式使用 CSV 文件并执行自定义分析。

CSV 文件包含以下列：

用户

用户的友好名称。

arn

用户的 Amazon Resource Name (ARN)。有关 ARN 的更多信息，请参阅 IAM ARN。

user_creation_time

创建用户的日期和时间，ISO 8601 日期时间格式。

password_enabled

如果用户有密码，则该值为 TRUE。否则为 FALSE。对于作为组织一部分创建的新成员账户，此值为 FALSE，因为默认情况下，这些账户没有根用户凭证。

password_last_used

Amazon Web Services 账户根用户 或用户的密码上次用于登录 Amazon 网站的日期和时间，采用 ISO 8601 日期时间格式。捕获用户上次登录时间的 Amazon 网站是 Amazon Web Services 管理控制台、Amazon 论坛和 Amazon Marketplace。如果密码在 5 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。

- 此字段的值在以下情况下为 no_information：
  - 用户的密码从未使用过。
  - 没有与密码关联的登录数据，例如，当用户的密码在 IAM 于 2014 年 10 月 20 日开始跟踪此信息后尚未使用时。
- 如果用户没有密码，此字段中的值为 N/A (不适用)。

Important

由于服务问题，上次使用密码的数据不包括从 2018 年 5 月 3 日 22:50（太平洋时间）到 2018 年 5 月 23 日 14:08（太平洋时间）之间的密码使用。这会影响 IAM 控制台中显示的上次登录日期和 IAM 凭证报告中以及 GetUser API 操作返回的上次使用密码的日期。如果用户在受影响的时间内登录，则返回的上次使用密码的日期是用户在 2018 年 5 月 3 日之前最后一次登录的日期。对于在 2018 年 5 月 23 日 14:08（太平洋时间）之后登录的用户，所返回的上次使用密码的日期准确无误。
如果您使用上次使用密码的信息来识别未使用的凭证以将其删除（如删除在过去 90 天内未登录过 Amazon 的用户），建议您调整评估时段以纳入 2018 年 5 月 23 日之后的日期。或者，

如果您的用户使用访问密钥以编程方式访问 Amazon，则您可以参阅上次使用访问密钥的信息，因为其中的所有日期都准确无误。

password_last_changed

上次设置用户密码的日期和时间，ISO 8601 日期时间格式。如果用户没有密码，则此字段中的值为 N/A (不适用)。

password_next_rotation

如果账户的密码策略要求密码轮换，则此字段包含用户需要设置新密码的日期和时间，ISO 8601 日期时间格式。Amazon Web Services 账户（根）的值始终是 not_supported。

mfa_active

如果对用户启用了多重验证 (MFA) 设备，则此值为 TRUE。否则为 FALSE。

access_key_1_active

如果用户有访问密钥且访问密钥状态为 Active，则该值为 TRUE。否则为 FALSE。适用于账户根用户和 IAM 用户。

access_key_1_last_rotated

创建或上次更改用户访问密钥的日期和时间，ISO 8601 日期时间格式。如果用户没有活动的访问密钥，则此字段中的值为 N/A (不适用)。适用于账户根用户和 IAM 用户。

access_key_1_last_used_date

最近使用用户的访问密钥签署 Amazon API 请求的日期和时间（采用 ISO 8601 日期时间格式）。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。

此字段的值在这些情况下为 N/A（不适用）：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 访问密钥在 IAM 于 2015 年 4 月 22 日开始跟踪此信息后尚未使用。

access_key_1_last_used_region

最近在其中使用访问密钥的 Amazon 区域。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。

此字段的值在这些情况下为 N/A（不适用）：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 上次使用访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。
- 上次使用的服务不是区域特定的服务，如 Amazon S3。

access_key_1_last_used_service

最近使用访问密钥访问的 Amazon 服务。此字段的值使用服务的命名空间，例如 s3 表示 Amazon S3，ec2 表示 Amazon EC2。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。

此字段的值在这些情况下为 N/A（不适用）：

- 用户没有访问密钥。
- 访问密钥从未使用过。
- 上次使用访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

access_key_2_active

如果用户有第二个访问密钥且第二个键的状态为 Active，则该值为 TRUE。否则为 FALSE。适用于账户根用户和 IAM 用户。

> **Note**
>
> 用户最多可以有两个访问密钥，为方便轮换，可以首先更新密钥，然后再删除之前的密钥。有关更新访问密钥的更多信息，请参阅 更新访问密钥。

access_key_2_last_rotated

创建或上次更改用户第二个访问密钥的日期和时间，采用 ISO 8601 日期时间格式。如果用户没有第二个活动的访问密钥，则此字段中的值为 N/A (不适用)。适用于账户根用户和 IAM 用户。

access_key_2_last_used_date

最近使用用户的第二个访问密钥签署 Amazon API 请求的日期和时间（采用 ISO 8601 日期时间格式）。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。

此字段的值在这些情况下为 N/A（不适用）：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

access_key_2_last_used_region

最近在其中使用用户的第二个访问密钥的 Amazon 区域。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。此字段的值在这些情况下为 N/A（不适用）：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。
- 上次使用的服务不是区域特定的服务，如 Amazon S3。

access_key_2_last_used_service

最近使用用户的第二个访问密钥访问的 Amazon 服务。此字段的值使用服务的命名空间，例如 s3 表示 Amazon S3，ec2 表示 Amazon EC2。如果访问密钥在 15 分钟的时间范围内多次使用，则仅在此字段中记录第一次使用。适用于账户根用户和 IAM 用户。此字段的值在这些情况下为 N/A（不适用）：

- 用户没有第二个访问密钥。
- 用户的第二个访问密钥从未使用过。
- 上次使用用户的第二个访问密钥是在 IAM 于 2015 年 4 月 22 日开始跟踪此信息之前。

cert_1_active

如果用户有 X.509 签名证书且证书状态为 Active，则该值为 TRUE。否则为 FALSE。

cert_1_last_rotated

创建或上次更改用户签名证书的日期和时间，ISO 8601 日期时间格式。如果用户没有活动的签名证书，则此字段中的值为 N/A (不适用)。

cert_2_active

如果用户有第二个 X.509 签名证书且证书状态为 Active，则该值为 TRUE。否则为 FALSE。

> **Note**
>
> 用户最多可以有两个 X.509 签名证书，以便于进行证书轮换。

cert_2_last_rotated

创建或上次更改用户第二个签名证书的日期和时间，ISO 8601 日期时间格式。如果用户没有第二个活动的签名证书，则此字段中的值为 N/A (不适用)。

additional_credentials_info

当用户拥有两个以上的访问密钥或证书时，该值是其他访问密钥或证书的数量，以及可用于列出与该用户关联的访问密钥或证书的操作。

获取凭证报告（控制台）

您可以使用 Amazon Web Services 管理控制台下载逗号分隔值 (CSV) 文件形式的凭证报告。

下载凭证报告（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择凭证报告。
3. 选择下载报告。

获取凭证报告 (Amazon CLI)

下载凭证报告 (Amazon CLI)

1. 生成凭证报告。Amazon 存储单个报告。如果存在报告，生成凭证报告将覆盖以前的报告。`aws iam generate-credential-report`
2. 查看上次生成的报告：`aws iam get-credential-report`

获取凭证报告 (Amazon API)

下载凭证报告 (Amazon API)

1. 生成凭证报告。Amazon 存储单个报告。如果存在报告，生成凭证报告将覆盖以前的报告。`GenerateCredentialReport`
2. 查看上次生成的报告：`GetCredentialReport`

CodeCommit 的 IAM 凭证：Git 凭证、SSH 密钥和 Amazon 访问密钥

CodeCommit 是一种在 Amazon 云中托管私有 Git 存储库的托管版本控制服务。要使用 CodeCommit，您需要配置 Git 客户端，使其与 CodeCommit 存储库通信。作为此配置的一部分，您需要提供 IAM 凭证，CodeCommit 将用该凭证来对您进行身份验证。IAM 支持具有三种凭证的 CodeCommit：

- Git 凭证 - 一种由 IAM 生成的用户名和密码对，供您通过 HTTPS 与 CodeCommit 存储库进行通信。
- SSH 密钥 - 一种本地生成的公有-私有密钥对，供您关联至 IAM 用户，以通过 SSH 与 CodeCommit 存储库进行通信。
- Amazon 访问密钥 - 供您配合 Amazon CLI 中包含的凭证辅助程序使用，以通过 HTTPS 与 CodeCommit 存储库通信。

> **Note**
>
> 您无法使用 SSH 密钥或 Git 凭证来访问另一个 Amazon 账户中的存储库。要了解如何在其他 Amazon Web Services 账户 中为 IAM 用户和组配置对 CodeCommit 存储库的访问权限，请参阅《Amazon CodeCommit 用户指南》使用角色配置对 Amazon CodeCommit 存储库的跨账户访问。

请参阅下面几节了解有关每个选项的更多信息。

将 Git 凭证和 HTTPS 与 CodeCommit 配合使用（推荐）

借助 Git 凭证，您可以为 IAM 用户生成静态用户名和密码对，然后使用这些凭证进行 HTTPS 连接。您还可以将这些凭证与支持静态 Git 凭证的任何第三方工具或集成开发环境 (IDE) 配合使用。

这些凭证对所有支持的操作系统来说是通用的，并且兼容大多数凭证管理系统、开发环境及其他软件开发工具，因此，这是我们推荐使用的方法。您可以随时重置 Git 凭证的密码。如果不再需要凭证，您还可以停用或删除它们。

> **Note**
>
> 您无法为 Git 凭证选择自己的用户名或密码。IAM 会为您生成这些凭证，以帮助确保它们符合 Amazon 的安全标准和 CodeCommit 中的安全存储库。凭证生成之后，您只能下载它们一次。

因此，请务必将凭证保存在安全的位置。如有必要，您可以随时重置密码，但这会使通过旧密码配置的所有连接失效。您必须使用新密码重新配置连接，然后才能再次进行连接。

请参阅以下主题了解更多信息：

- 要创建 IAM 用户，请参阅 在 Amazon Web Services 账户 中创建 IAM 用户。
- 要借助 CodeCommit 生成并使用 Git 凭证，请参阅 Amazon CodeCommit 用户指南中的适用于使用 Git 凭证的 HTTPS 用户。

> **Note**
>
> 在生成 Git 凭证后更改 IAM 用户的名称不会更改 Git 凭证的用户名。用户名和密码保持不变且仍然有效。

更新服务特定凭证

1. 除了当前正在使用的一组凭证外，创建第二组特定于服务的凭证。
2. 更新所有应用程序以使用这组新凭证，然后验证应用程序能否正常工作。
3. 将原始凭证的状态更改为“停用”。
4. 确保您的所有应用程序仍能正常工作。
5. 删除已停用的特定于服务的凭证。

借助 CodeCommit 使用 SSH 密钥和 SSH

借助 SSH 连接，您可以在本地计算机上创建公有和私有密钥文件，以供 Git 和 CodeCommit 进行 SSH 身份验证。将公有密钥关联至 IAM 用户，并将私有密钥存储在本地计算机上。请参阅以下主题了解更多信息：

- 要创建 IAM 用户，请参阅 在 Amazon Web Services 账户 中创建 IAM 用户。
- 要创建 SSH 公有密钥并将其与 IAM 用户相关联，请参阅 Amazon CodeCommit 用户指南中的适用于 Linux、macOS 或 Unix 上的 SSH 连接或请参阅适用于 Windows 上的 SSH 连接。

> **Note**
>
> 公有密钥必须采用 ssh-rsa 格式或 PEM 格式进行编码。公有密钥的最小位长度为 2048 位，最大长度为 16384 位。这独立于您上传的文件大小。例如，您可以生成 2048 位密钥，但生成的 PEM 文件大小为 1679 字节。如果您采用其他格式或大小提供公有密钥，则系统会显示一条错误消息，指出该密钥格式无效。

将 HTTPS 与 Amazon CLI 凭证辅助程序和 CodeCommit 配合使用

作为使用 Git 凭证进行 HTTPS 连接的替代方法，您可以允许 Git 在需要进行 Amazon 身份验证以与 CodeCommit 存储库交互时使用 IAM 用户凭证的加密签名版本或 Amazon EC2 实例角色。这是唯一一种不需要 IAM 用户就能连接 CodeCommit 存储库的方法。这也是唯一一种适用于联合访问和临时凭证的方法。请参阅以下主题了解更多信息：

- 有关联合访问的更多信息，请参阅身份提供程序和 Amazon 中的联合身份验证和访问经过外部身份验证的用户（身份联合验证）。
- 有关临时凭证的更多信息，请参阅 IAM 临时安全凭证 和临时访问 CodeCommit 存储库。

Amazon CLI 凭证辅助程序不兼容 Keychain Access、Windows Credential Management 等其他凭证辅助系统。在使用凭证辅助程序配置 HTTPS 连接时，还有一些其他的配置注意事项。有关更多信息，请参阅 Amazon CodeCommit 用户指南中的使用 Amazon CLI 凭证辅助程序在 Linux、macOS 或 Unix 上进行 HTTPS 连接或使用 Amazon CLI 凭证辅助程序在 Windows 上进行 HTTPS 连接。

在 IAM 中管理服务器证书

要在 Amazon 中启用与您的网站或应用程序的 HTTPS 连接，您需要 SSL/TLS 服务器证书。对于 Amazon Certificate Manager (ACM) 支持的区域中的证书，我们建议您使用 ACM 预置、管理和部署您的服务器证书。在不支持的区域中，您必须将 IAM 作为证书管理器。要了解 ACM 支持的具体区域，请参阅《Amazon Web Services 一般参考》中的 Amazon Certificate Manager 端点和限额。

只有当您必须在 ACM 不支持的区域中支持 HTTPS 连接时，才应使用 IAM 作为证书管理器。IAM 安全地加密您的私有密钥并将加密的版本存储在 IAM SSL 证书存储中。IAM 支持在所有区域部署服务器证书，但您必须从外部提供商获取证书，以便与 Amazon 搭配使用。您无法将 ACM 证书上传到 IAM。此外，您还无法从 IAM 控制台管理证书，

有关将第三方证书上传到 IAM 的更多信息，请参阅以下主题。

主题

- 上传服务器证书（Amazon API）
- 服务器证书的 Amazon API 操作
- 排查服务器证书问题

上传服务器证书（Amazon API）

要将服务器证书上传到 IAM，您必须提供证书及其匹配的私有密钥。如果证书不是自签名的，则您还必须提供证书链。(上传自签名证书时无需证书链。) 在上传证书前，请确保您已具有所有这些项目而且它们满足以下条件：

- 证书在上传时必须是有效的。您不能在证书有效期开始 (证书的 `NotBefore` 日期) 之前或证书有效期到期 (证书的 `NotAfter` 日期) 之后上传证书。
- 私有密钥必须是未加密的。您不能上传受密码或口令保护的私有密钥。有关解密已加密的私有密钥的帮助信息，请参阅排查服务器证书问题。
- 证书、私有密钥和证书链必须均采用 PEM 编码。有关将这些项目转换为 PEM 格式的帮助信息，请参阅排查服务器证书问题。

要使用 IAM API 上传证书，请发送 UploadServerCertificate 请求。以下示例说明如何使用 Amazon Command Line Interface (Amazon CLI) 执行该操作。示例假定以下各项：

- PEM 编码的证书存储在名为 `Certificate.pem` 的文件中。
- PEM 编码的证书链存储在名为 `CertificateChain.pem` 的文件中。
- PEM 编码的未加密私有密钥存储在名为 `PrivateKey.pem` 的文件中。
- （可选）您希望用键-值对标记服务器证书。例如，您可以添加标签键 `Department` 和标签值 `Engineering` 来帮助识别和组织证书。

要使用以下示例命令，请用自己的文件名替换这些文件名。将 *`ExampleCertificate`* 替换为已上传证书的名称。如果要标记证书，请将 *`ExampleKey`* 和 *`ExampleValue`* 标签键值对替换为您自己的值。在一个连续行上键入命令。为更便于阅读，以下示例包含了换行符和多余的空格。

```
aws iam upload-server-certificate --server-certificate-name ExampleCertificate
                                  --certificate-body file://Certificate.pem
                                  --certificate-chain file://CertificateChain.pem
                                  --private-key file://PrivateKey.pem
```

```
                                    --tags '{"Key": "ExampleKey", "Value":
"ExampleValue"}'
```

如果上述命令执行成功，则它将返回有关上传的证书的元数据，包括其 Amazon Resource Name (ARN)、友好名称、标识符 (ID)、到期日期、标签等。

> **Note**
>
> 如果您要上传服务器证书以用于 Amazon CloudFront，则必须使用 `--path` 选项指定路径。路径必须以 `/cloudfront` 开头且必须包含尾部反斜杠 (例如，`/cloudfront/test/`)。

要使用 Amazon Tools for Windows PowerShell上传证书，请使用 Publish-IAMServerCertificate。

服务器证书的 Amazon API 操作

使用以下命令来查看、标记、重命名和删除服务器证书。

- 使用 GetServerCertificate 命令来检索证书。此请求会返回证书、证书链（如果已上传一个）和有关证书的元数据。

  > **Note**
  >
  > 在您上传后，无法从 IAM 下载或检索私有密钥。

- 使用 Get-IAMServerCertificate 来检索证书。
- 使用 ListServerCertificates 列出已上传的服务器证书。该请求会返回包含有关每个证书的元数据的列表。
- 使用 Get-IAMServerCertificates 列出已上传的服务器证书。
- 使用 TagServerCertificate 标记现有的服务器证书。
- 使用 UntagServerCertificate 取消标记服务器证书。
- 使用 UpdateServerCertificate 重命名服务器证书或更新其路径。

  以下示例说明如何使用 Amazon CLI 执行该操作。

  要使用以下示例命令，请将旧的证书名称和证书路径替换为新的，然后在一个连续行上键入命令。为更便于阅读，以下示例包含了换行符和多余的空格。

```
aws iam update-server-certificate --server-certificate-name ExampleCertificate
                                  --new-server-certificate-
name CloudFrontCertificate
                                  --new-path /cloudfront/
```

要使用 Amazon Tools for Windows PowerShell 重命名服务器证书或更新其路径，请使用 Update-IAMServerCertificate。

- 使用 DeleteServerCertificate 删除服务器证书。

  要使用 Amazon Tools for Windows PowerShell 删除服务器证书，请使用 Remove-IAMServerCertificate。

排查服务器证书问题

您必须先确保证书、私有密钥和证书链均是 PEM 编码的，然后才能将证书上传到 IAM。您还必须确保私有密钥是未加密的。请见以下 示例。

Example PEM 编码证书示例

```
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

Example PEM 编码的未加密的私有密钥示例

```
-----BEGIN RSA PRIVATE KEY-----
Base64-encoded private key
-----END RSA PRIVATE KEY-----
```

Example PEM 编码的证书链示例

一个证书链包含一个或多个证书。您可以使用文本编辑器、Windows 中的复制命令或 Linux cat 命令，以将证书文件串联到一个链中。如果包含多个证书，则每个证书必须证明上一个证书。您可以串联证书以完成该操作，包括最后的根 CA 证书。

以下示例包含三个证书，但证书链可能包含更多或更少的证书。

```
-----BEGIN CERTIFICATE-----
```

```
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
Base64-encoded certificate
-----END CERTIFICATE-----
```

如果这些项目在上传到 IAM 时未采用正确的格式，您可以使用 OpenSSL 将其转换为正确的格式。

将证书或证书链从 DER 转换为 PEM

使用 OpenSSL x509 命令，如以下示例所示。在以下示例命令中，将 *Certificate.der* 替换为包含您的 DER 编码的证书的文件的名称。将 *Certificate.pem* 替换为要包含 PEM 编码的证书的输出文件的首选名称。

```
openssl x509 -inform DER -in Certificate.der -outform PEM -out Certificate.pem
```

将私有密钥从 DER 转换为 PEM

使用 OpenSSL rsa 命令，如以下示例所示。在以下示例命令中，将 *PrivateKey.der* 替换为包含您的 DER 编码的私有密钥的文件的名称。将 *PrivateKey.pem* 替换为要包含 PEM 编码的私有密钥的输出文件的首选名称。

```
openssl rsa -inform DER -in PrivateKey.der -outform PEM -out PrivateKey.pem
```

解密已加密的私有密钥 (删除密码或口令)

使用 OpenSSL rsa 命令，如以下示例所示。要使用以下示例命令，请将 *EncryptedPrivateKey.pem* 替换为包含您的已加密的私有密钥的文件的名称。将 *PrivateKey.pem* 替换为要包含 PEM 编码的未加密私有密钥的输出文件的首选名称。

```
openssl rsa -in EncryptedPrivateKey.pem -out PrivateKey.pem
```

将证书包从 PKCS#12 (PFX) 转换为 PEM

使用 OpenSSL pkcs12 命令，如以下示例所示。在以下示例命令中，将 *CertificateBundle.p12* 替换为包含您的 PKCS#12 编码的证书包的文件的名称。将 *CertificateBundle.pem* 替换为要包含 PEM 编码的证书捆绑包的输出文件的首选名称。

```
openssl pkcs12 -in CertificateBundle.p12 -out CertificateBundle.pem -nodes
```

将证书包从 PKCS#7 转换为 PEM

使用 OpenSSL pkcs7 命令，如以下示例所示。在以下示例命令中，将 *CertificateBundle.p7b* 替换为包含您的 PKCS#7 编码的证书包的文件的名称。将 *CertificateBundle.pem* 替换为要包含 PEM 编码的证书捆绑包的输出文件的首选名称。

```
openssl pkcs7 -in CertificateBundle.p7b -print_certs -out CertificateBundle.pem
```

IAM 用户组

IAM 用户组是 IAM 用户的集合。利用用户组，可为多个用户指定权限，以便更轻松地管理这些用户的权限。例如，您可能有一个名为 Admins 的用户组，并向该用户组授予管理员通常需要的权限类型。该用户组中的所有用户均自动拥有 Admins 组权限。如果有新用户加入您的组织，并且需要管理员权限，则可通过将此用户添加到 Admins 用户组来分配相应的权限。如果组织中有成员更换岗位，则不必编辑该用户的权限，只需从旧 IAM 组中将此用户删除，然后将其添加到相应的新 IAM 组即可。

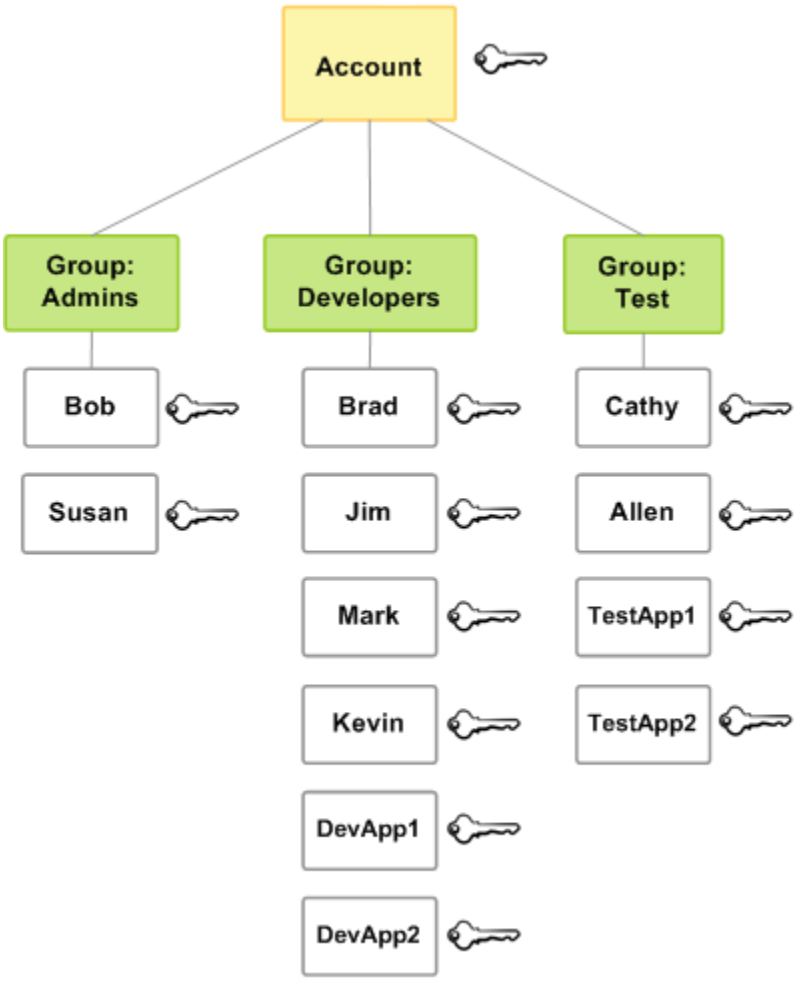
您可以将基于身份的策略附加到用户组，以便该用户组中的所有用户都能获得该策略的权限。您无法在策略（例如基于资源的策略）中将用户组标识为 `Principal`，因为组与权限相关，与身份验证无关，并且主体是经过身份验证的 IAM 实体。有关策略类型的更多信息，请参阅基于身份的策略和基于资源的策略。

以下为 IAM 组具有的一些重要特点：

- 一个用户组可包含多个用户，而一个用户又可归属于多个用户组。
- 用户组不能嵌套；其中只能包含用户，而不能包含其他 IAM 组。
- 默认情况下，没有可自动包含 Amazon Web Services 账户 内所有用户的用户组。如果希望有这样的用户组，则需要创建该组，然后将每个新用户分配给它。

- Amazon Web Services 账户 中 IAM 资源的数量和大小（例如组的数量和用户可加入的组的数量）是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

下图以一家小型公司作为简单示例。随着公司的发展，公司所有者为用户创建了一个 Admins 用户组来创建和管理其他用户。Admins 用户组会创建一个 Developers 用户组和 Test 用户组。这两个 IAM 组都包含可与 Amazon 交互的用户（Jim、Brad、DevApp1 等）。每个用户均拥有一组单独的安全证书。在此示例中，每个用户均属于单一用户组。不过，用户可归属于多个 IAM 组。



创建 IAM 组

> **Note**
>
> 作为最佳实践，我们建议您要求人类用户使用带有身份提供程序的联合身份验证才能使用临时凭证访问 Amazon。如果您遵循最佳实践，则无法管理 IAM 用户和组。相反，您的用户和组是在 Amazon 外部进行管理的，并且能够以联合身份访问 Amazon 资源。联合身份是来自企业

用户目录、Web 身份提供程序、Amazon Identity Service 的用户，或任何使用通过身份源提供的凭证来访问 Amazon 服务的用户。联合身份使用其身份提供商定义的组。如果您使用的是 Amazon IAM Identity Center，请参阅《Amazon IAM Identity Center 用户指南》中的管理 IAM Identity Center 中的身份，了解有关在 IAM Identity Center 中创建用户和组的信息。

您可以创建 IAM 组，以管理具有相似角色或职责的多个用户的访问权限。您可以通过将策略附加到这些组来授予或撤消整组用户的权限。您对组的权限所做的更改将自动应用于该组的所有成员，以确保一致的访问控制，因此这简化了您对安全策略的维护。创建组后，根据您希望组中的 IAM 用户执行的工作类型向组授予权限，然后将 IAM 用户添加到该组。

有关创建 IAM 组所需权限的信息，请参阅访问 IAM 资源所需的权限。

创建 IAM 组并附加策略

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 User groups（用户组），然后选择 Create group（创建组）。
3. 对于 User group name（用户组名称），键入组的名称。

   > **Note**
   >
   > Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。组名称可以是一个最多由 128 个字母、数字和以下字符构成的组合：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、at 符号 (@)、下划线 (_) 和连字符 (-)。账户中的名称必须唯一。名称不区分大小写。例如，您无法同时创建名为 **ADMINS** 和 **admins** 的组。

4. 在用户列表中，选中要添加到组中的每个用户对应的复选框。
5. 在策略列表中，选中要应用于组中的所有成员的每个策略的复选框。
6. 选择创建群组。

Amazon CLI

运行以下命令：

- aws iam create-group

API

调用以下操作：

- CreateGroup

查看 IAM 组

您可以列出账户中的所有 IAM 组、列出某个用户组中的用户或列出用户所属的 IAM 组。如果您使用 CLI 或 API，则可以列出带特定路径前缀的所有 IAM 组。

Console

列出账户中的所有 IAM 组：

- 在导航窗格中，选择用户组。

列出特定 IAM 组中的 IAM 用户：

- 在导航窗格中，选择 User groups（用户组）。选择组的名称，打开组详细信息页面。查看用户选项卡可查看组成员资格。

列出用户所属的所有 IAM 组：

- 在导航窗格中，选择 Users（用户）。然后选择用户名称，以打开用户详细信息页面。选择组选项卡可查看用户所属组的列表。

Amazon CLI

列出账户中的所有 IAM 组：

- aws iam list-groups

列出特定 IAM 组中的用户：

- aws iam get-group

列出用户所属的所有 IAM 组：

- aws iam list-groups-for-user

API

列出账户中的所有 IAM 组：

- ListGroups

列出特定 IAM 组中的用户：

- GetGroup

列出用户所属的所有 IAM 组：

- ListGroupsForUser

编辑 IAM 组中的用户

使用 IAM 组可以跨多个有用户同时应用相同的权限策略。然后，您可以在 IAM 组中添加或删除用户。由于您组织的人员是流动的，因此这样做会很有用。

检查策略访问权限

在删除组之前，可利用组详细信息页面在权限选项卡上查看组成员（IAM 用户）和附加到组的策略，并使用上次访问时间选项卡查看最近的服务级别活动。这有助于防止意外删除正在使用访问权限的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

将 IAM 用户添加到 IAM 组

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，请选择 User groups（用户组），然后选择组的名称。

3. 请选择 Users（用户）选项卡，然后选择 Add users（添加用户）。选中要添加的用户旁边的复选框。
4. 选择 Add Users（添加用户）。

Amazon CLI

运行以下命令：

- `aws iam add-user-to-group`

API

调用以下操作：

- `AddUserToGroup`

从 IAM 组中删除 IAM 用户

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，请选择 User groups（用户组），然后选择组的名称。
3. 选择用户选项卡。选中要移除的用户旁边的复选框，然后选择 Remove users（移除用户）。

Amazon CLI

运行以下命令：

- `aws iam remove-user-from-group`

API

调用以下操作：

- `RemoveUserFromGroup`

将策略附加到 IAM 用户组

您可以将 Amazon 托管策略 - 由 Amazon 提供的预先写入的策略 - 附加到用户组，如下列步骤中所述。要附加客户托管策略（即具有您创建的自定义权限的策略），必须首先创建策略。有关创建客户托管策略的信息，请参阅使用客户管理型策略定义自定义 IAM 权限。

有关权限和策略的更多信息，请参阅适用于 Amazon 资源的 Access Management。

将策略附加到 IAM 组

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，请选择 User groups（用户组），然后选择组的名称。
3. 选择 Permissions（权限）选项卡。
4. 选择添加权限，然后选择附加策略。
5. 附加到用户组的当前策略显示在 Current permissions policies（当前权限策略）列表。在 Other permissions policies（其他权限策略）列表中，选中要附加的策略名称旁边的复选框。您可以使用搜索框通过输入类型和策略名称来筛选策略列表。
6. 选择您要附加到 IAM 组的策略，然后选择附加策略。

Amazon CLI

运行以下命令：

- `aws iam attach-group-policy`

API

调用以下操作：

- `AttachGroupPolicy`

重命名 IAM 用户组

当您更改用户组的名称或路径时，发生以下情况：

- 应用于用户组的所有策略采用新群组名称继续生效。
- 该用户组采用新名称保留所有用户。
- 该用户组的唯一 ID 保持不变。有关唯一 ID 的更多信息，请参阅 唯一标识符。

IAM 不自动更新将该用户组作为资源引用的策略以使用新名称。因此，重命名用户组时，请务必保持谨慎。在重命名用户组之前，您必须手动检查所有策略以按名称查找提到该用户组的任何策略。例如，假设 Bob 是公司测试部门的经理。Bob 向他的 IAM 用户实体中挂载了一个策略，该策略允许他在“测试”用户组中添加和删除用户。如果管理员更改用户组的名称（或更改组的路径），则管理员还必须更新附加到 Bob 的策略以使用新名称或新路径。否则，Bob 将无法在该用户组中添加或删除用户。

查找将 IAM 组作为资源引用的策略：

1. 从 IAM 控制台的导航窗格中，选择 Policies（策略）。
2. 按 Type（类型）列排序，以查找您的 Customer managed（客户托管）自定义策略。
3. 请选择策略的名称进行编辑。
4. 选择权限选项卡，然后选择摘要。
5. 从服务列表中选择 IAM（如果存在）
6. 在资源列中查找您的用户组名称。
7. 选择编辑以在策略中更改您的用户组名称。

更改 IAM 用户组的名称

Console

1. 在导航窗格中，选择 用户组，然后选择组的名称。
2. 选择编辑。键入新的用户组名称，然后选择 Save changes（保存更改）。

Amazon CLI

运行以下命令：

- aws iam update-group

API

调用以下操作：

- UpdateGroup

删除 IAM 组

当您在控制台中删除 IAM 组时，控制台将自动删除所有组成员、分离所有已附加的托管策略并删除所有内联策略。不过，IAM 不会自动删除将 IAM 组作为资源引用的策略，因此，在删除 IAM 组时必须格外小心。在删除 IAM 组之前，您必须手动检查策略，按名称查找提到该组的任何策略。例如，测试团队负责人 John 向其 IAM 用户实体附加了一个策略，该策略允许他在“测试”用户组中添加和删除用户。如果管理员删除该组，管理员还必须删除挂载到 John 的策略。否则，如果管理员重新创建已删除组并以相同名称命名，即使 John 已经离开测试团队，其权限仍然保持不变。

相反，在使用 CLI、SDK 或 API 删除组时，必须先删除组中的用户。然后删除嵌入到 IAM 组中的所有内联策略。接下来，分离所有已附加到组的托管策略。只有这样才能删除 IAM 组本身。

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 User groups（用户组）。
3. 在 IAM 组列表中，选中要删除的 IAM 组的名称旁边的复选框。您可以使用搜索框按类型、权限和组名称筛选 IAM 组列表。
4. 选择删除。
5. 在确认框中，如果要删除单个组，请键入组名称，然后选择删除。如果要删除多个组，请键入要删除的 IAM 组数，然后键入 **`user groups`**，并选择删除。例如，若要删除三个组，则键入 **`3 user groups`**。

Amazon CLI

1. 从 IAM 组移除所有用户。

   - aws iam get-group（用于获取 IAM 组中的用户列表）和 aws iam remove-user-from-group（用于从 IAM 组中删除用户）

2. 删除嵌入到 IAM 组中的所有内联策略。

   - aws iam list-group-policies（用于获取 IAM 组的内联策略的列表）和 aws iam delete-group-policy（用于删除 IAM 组的内联策略）

3. 分离所有已附加到 IAM 组的托管策略。

- aws iam list-attached-group-policies（用于获取已附加到 IAM 组的托管式策略的列表）和 aws iam detach-group-policy（用于将托管式策略从 IAM 组中分离）

4. 删除 IAM 组。

   - aws iam delete-group

API

1. 从 IAM 组移除所有用户。

   - GetGroup（用于获取 IAM 组中的用户的列表）和 RemoveUserFromGroup（用于从 IAM 组中删除用户）

2. 删除嵌入到 IAM 组中的所有内联策略。

   - ListGroupPolicies（用于获取 IAM 组的内联策略的列表）和 DeleteGroupPolicy（用于删除 IAM 组的内联策略）

3. 分离所有已附加到 IAM 组的托管策略。

   - ListAttachedGroupPolicies（用于获取已附加到 IAM 组的托管式策略的列表）和 DetachGroupPolicy（用于将托管式策略从 IAM 组中分离）

4. 删除 IAM 组。

   - DeleteGroup

IAM 角色

IAM 角色是可在账户中创建的一种具有特定权限的 IAM 身份。IAM 角色类似于 IAM 用户，因为它是一个 Amazon 身份，具有确定其在 Amazon 中可执行和不可执行的操作的权限策略。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

您可以使用角色向通常无权访问您的 Amazon 资源的用户、应用程序或服务提供访问权限。例如，您可能需要向您 Amazon 账户中的用户授予对其通常不拥有的资源的访问权限，或是向一个 Amazon Web Services 账户 中的用户授予对其他账户中的资源的访问权限。或者，您可能需要允许移动应用程序使用 Amazon 资源，但是不希望在应用程序中嵌入 Amazon 密钥（因为难以更新密钥，而且用户有可能提取到密钥）。有时，您需要向已经具有在 Amazon 外部（例如，在您的公司目录中）定义的身

份的用户提供对 Amazon 的访问权限。或者，您可能需要向第三方授予对您账户的访问权限，使他们可以对您的资源执行审核。

对于这些情况，您可以使用 IAM 角色委派对 Amazon 资源的访问权限。本节介绍各种角色和它们的不同使用方式，何时及如何选择方法，如何创建、管理、切换到（或担任）和删除角色。

> **Note**
>
> 首次创建您的 Amazon Web Services 账户 时，默认情况下不会创建任何角色。当您向账户添加服务时，这些服务可能会添加服务相关角色以支持其使用案例。
> 服务关联角色是一种与 Amazon Web Services 服务 关联的服务角色。服务可以代入代表您执行操作的角色。服务关联角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务关联角色的权限。
> 您必须先删除角色的相关资源，之后才能删除服务相关角色。这可以保护您的资源，因为您不会无意中删除对资源的访问权限。
> 有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

主题

- 何时创建 IAM 用户（而不是角色）
- 角色术语和概念
- 其他资源
- 混淆代理人问题
- IAM 角色的常见场景
- IAM 角色创建
- IAM 角色管理
- 担任角色的方法

何时创建 IAM 用户（而不是角色）

我们建议您仅在身份联合验证不支持的使用案例中使用 IAM 用户。部分使用场景包括：

- 无法使用 IAM 角色的工作负载 – 您可以从需要访问 Amazon 的位置运行工作负载。在某些应用场景中，您无法使用 IAM 角色提供临时凭证，例如 WordPress 插件。在这些情况下，请使用该工作负载的 IAM 用户长期访问密钥对 Amazon 进行身份验证。

- 第三方 Amazon 客户端：如果您使用的工具不支持访问 IAM Identity Center，例如不在 Amazon 上托管的第三方 Amazon 客户端或供应商，请使用 IAM 用户长期访问密钥。
- Amazon CodeCommit 访问 – 如果您正在使用 CodeCommit 来存储您的代码，则可以使用具有 SSH 密钥或服务特定凭证的 IAM 用户对 CodeCommit 进行存储库身份验证。我们建议您除了使用 IAM Identity Center 中的用户进行普通身份验证之外，再执行此操作。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon Web Services 账户 或云应用程序的人员。要在不配置 IAM 用户的情况下授予用户访问您 CodeCommit 存储库的权限，您可以配置 git-remote-codecommit 实用工具。有关 IAM 和 CodeCommit 的更多信息，请参阅 CodeCommit 的 IAM 凭证：Git 凭证、SSH 密钥和 Amazon 访问密钥。有关配置 git-remote-codecommit 实用工具的更多信息，请参阅《Amazon CodeCommit 用户指南》中的连接到具有轮换凭证的 Amazon CodeCommit 存储库。
- Amazon Keyspaces (for Apache Cassandra) access（Amazon Keyspaces（Apache Cassandra 兼容）访问）– 在您无法使用 IAM Identity Center 中的用户的情况下，如出于测试 Cassandra 兼容性的目的，您可以使用具有服务特定凭证的 IAM 用户向 Amazon Keyspaces 进行身份验证。IAM Identity Center 中的用户是您的员工队伍中需要访问您的 Amazon Web Services 账户 或云应用程序的人员。您还可以使用临时凭证连接到 Amazon Keyspaces。有关更多信息，请参阅《Amazon Keyspaces（Apache Cassandra 兼容）开发人员指南》中的使用临时凭证连接到使用 IAM 角色和 SigV4 插件的 Amazon Keyspaces。
- 紧急访问：无法访问身份提供者，且必须在 Amazon Web Services 账户 中执行操作的情况。创建紧急访问 IAM 用户可以是您弹性计划的一部分。我们建议使用多重身份验证（MFA）严格控制和保护紧急用户凭证。

角色术语和概念

以下是可帮助您开始使用角色的一些基本术语。

角色

可在账户中创建的具有特定权限的 IAM 身份。IAM 角色与 IAM 用户有一些相似之处。角色和用户都是具有权限策略的 Amazon 身份，该策略可确定身份在 Amazon 中可执行和不可执行的操作。但是，角色旨在让需要它的任何人代入，而不是唯一地与某个人员关联。此外，角色没有关联的标准长期凭证（如密码或访问密钥）。相反，当您代入角色时，它会为您提供角色会话的临时安全凭证。

角色可由以下用户代入：

- 相同 Amazon Web Services 账户 或另一个 Amazon Web Services 账户 中的 IAM 用户
- 同一账户中的 IAM 角色

- 服务主体，用于 Amazon 服务和功能，例如：
  - 允许您在计算服务上运行代码的服务，如 Amazon EC2 或 Amazon Lambda
  - 代表您对资源执行操作的功能，如 Amazon S3 对象复制
  - 为在 Amazon 外部运行的应用程序提供临时安全凭证的服务，如 IAM Roles Anywhere 或 Amazon ECS Anywhere
- 由与 SAML 2.0 或 OpenID Connect 兼容的外部身份提供者（IdP）服务进行身份验证的外部用户

Amazon 服务 角色

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务 委派权限的角色。

Amazon 服务相关角色

服务关联角色是一种与 Amazon Web Services 服务 关联的服务角色。服务可以代入代表您执行操作的角色。服务关联角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

> **Note**
>
> 如果在某项服务开始支持服务相关角色时您已使用该服务，您可能会收到一封电子邮件，告知您账户中的新角色。在这种情况下，该服务已自动在您的账户中创建服务相关角色。您无需执行任何操作来支持该角色，并且您不应手动删除它。有关更多信息，请参阅 我的 Amazon 账户中出现新角色。

有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。请选择是与查看该服务的服务关联角色文档的链接。有关更多信息，请参阅 创建服务相关角色。

角色链

角色链是指使用一个角色代入另一个角色。您可以使用 Amazon Web Services 管理控制台 通过切换角色、Amazon CLI 或 API 来执行角色链。例如，`RoleA` 有权代入 `RoleB`。您可以在 AssumeRole API 操作中使用其长期用户凭证以允许 User1 代入 `RoleA`。该操作会返回 `RoleA` 短期凭证。借助角色链，您可以使用 `RoleA` 的短期凭证允许 User1 代入 `RoleB`。

当您在代入某个角色的同时，您可以传递会话标签并将键设置为可传递。可传递会话标签将传递给角色链中的所有后续会话。要了解会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。

角色链将您的 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 角色会话限制为最长 1 小时。无论为各个角色配置的最长会话持续时间如何，此限制均适用。在使用 AssumeRole API 操作代入角色时，您可以使用 DurationSeconds 参数指定角色会话的持续时间。您可以指定最多为 43200 秒 (12 小时) 的参数值，具体取决于您的角色的最大会话持续时间设置。不过，如果您使用角色链代入角色并提供大于 1 小时的 DurationSeconds 参数值，则操作将失败。

有关在 Amazon Web Services 管理控制台 中切换到某个角色的信息，请参阅 从用户切换到 IAM 角色（控制台）。

委派

为某人授予权限以允许访问您控制的资源。委派涉及在两个账户之间建立信任。第一个是拥有资源的账户（信任账户）。第二个是包含需要访问资源的用户的账户（可信账户）。可信账户和信托账户可为以下任一账户：

- 同一账户。
- 受您的组织控制的两个单独账户。
- 由不同组织拥有的两个账户。

要委派权限以访问资源，请在附加了两个策略的信任账户中创建 IAM 角色。权限策略授予角色的用户所需权限对资源执行预期任务。信任策略指定允许哪些受信任账户成员担任角色。

创建信任策略时，不得在主体元素中指定通配符（*）作为 ARN 的一部分。信任策略附加到信任账户中的角色，并且是权限的一半。另一半是附加到允许用户切换到或担任角色的受信任账户中的用户的权限策略。临时担任一个角色的用户将放弃自己的权限而接过该角色的权限。用户退出或停止使用角色时，恢复原始用户权限。另一个名为外部 ID 的参数有助于确保在不受同一企业控制的账户之间安全使用角色。

信任策略

在 JSON 策略文档中，您可以定义您信任代入该角色的主体。角色信任策略是必需的基于资源的策略（将附加到 IAM 中的角色）。您可以在信任策略中指定的主体包括用户、角色、账户和服务。有关更多信息，请参阅 Amazon 安全博客中的如何在 IAM 角色中使用信任策略。

用于跨账户访问的角色

将一个账户中的资源的访问权限授予另一个账户中的受信任主体的账户。角色是授予跨账户访问权限的主要方式。但是，某些 Amazon 服务允许您将策略直接附加到资源（而不是使用角色作为代理）。这些策略称为基于资源的策略，您可以使用它们向其他 Amazon Web Services 账户中的主体授予对资源的访问权限。其中一些资源包括 Amazon Simple Storage Service (S3) 存储桶、Amazon Glacier 文件库、Amazon Simple Notification Service (SNS) 主题和 Amazon Simple

Queue Service (SQS) 队列。要了解哪些服务支持基于资源的策略，请参阅 使用 IAM 的 Amazon 服务。有关基于资源的策略的更多信息，请参阅 IAM 中的跨账户资源访问。

其他资源

以下资源可帮助您了解与 IAM 角色相关的 IAM 术语的更多信息。

- 主体是 Amazon 中可执行操作并访问资源的实体。主体可以是 Amazon Web Services 账户根用户、IAM 用户或角色。代表 Amazon 服务身份的主体就是服务主体。使用角色信任策略中的 Principal 元素来定义您信任的可以代入该角色的主体。

  有关您可以允许代入角色的主体的更多信息和示例，请参阅 Amazon JSON 策略元素：Principal。
- 身份联合验证会创建外部身份提供商与 Amazon 之间的信任关系。您可以使用现有的 OpenID Connect（OIDC）或安全断言标记语言（SAML）2.0 提供程序来管理谁可以访问 Amazon 资源。当您使用 OIDC 和 SAML 2.0 在这些外部身份提供者与 Amazon 之间配置信任关系时，会将用户分配到 IAM 角色。用户还会获得允许该用户访问您的 Amazon 资源的临时凭证。

  有关联合主体的更多信息，请参阅 身份提供程序和 Amazon 中的联合身份验证。
- 联合主体是来自 Amazon Directory Service、您的企业用户目录或 OIDC 提供者的现有身份。Amazon 在通过身份提供者请求访问权限时，将为联合主体分配角色。

  有关 SAML 和 OIDC 联合主体的更多信息，请参阅 联合用户会话和角色。
- 权限策略是基于身份的策略，用于定义角色可使用的操作和资源。该文档是根据 IAM policy 语言的规则编写的。

  有关更多信息，请参阅 IAM JSON 策略参考。
- 权限边界是一项高级功能，利用该功能，可以使用策略来限制基于身份的策略可以授予角色的最大权限。无法将权限边界应用于服务相关角色。

  有关更多信息，请参阅 IAM 实体的权限边界。

混淆代理人问题

混淆代理问题是一个安全问题，即没有执行操作权限的实体可能会迫使更具权限的实体执行该操作。为了防止这种情况，如果您为账户中的资源提供第三方（称为跨账户）或其他 Amazon 服务（称为跨服务）的访问权限，则 Amazon 会提供用于保护您账户的工具。

有时，您需要向第三方提供对您的 Amazon 资源的访问权 (提供访问权)。举例来说，您决定聘请一家名为 Example Corp 的第三方公司来监控您的 Amazon Web Services 账户并帮助优化成本。为跟踪您的日常开支，Example Corp 需要访问您的 Amazon 资源。Example Corp 也可监控其他客户的很多其他 Amazon Web Services 账户。您可使用 IAM 角色在您的 Amazon Web Services 账户 和 Example Corp 账户之间建立信任关系。此方案的一个重要方面是外部 ID，外部 ID 是一个可选标识符，您可在 IAM 角色信任策略中使用该信息来指定谁能代入该角色。外部 ID 的主要功能是解决并防止混淆代理人问题。

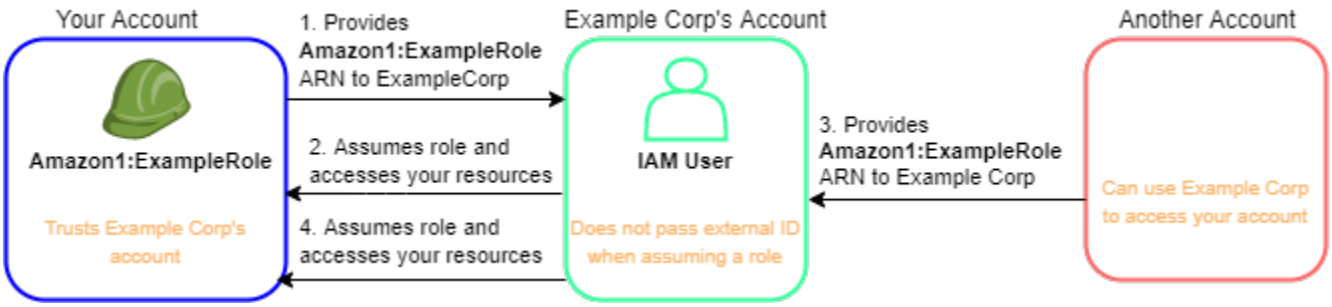
某些 Amazon 服务（呼叫服务）使用其 Amazon 服务主体从其他 Amazon 服务（被叫服务）访问 Amazon 资源。在其中一些服务交互中，您可以将呼叫服务配置为与来自其他 Amazon Web Services 账户中的被叫服务的资源进行通信。例如，配置 Amazon CloudTrail 以写入位于不同 Amazon Web Services 账户的中央 Amazon S3 存储桶。通过为 `cloudtrail.amazonaws.com` 添加允许声明，调用服务 CloudTrail 可以使用 S3 存储桶的策略访问您的 S3 存储桶。

当来自呼叫服务的 Amazon 服务主体访问来自被叫服务的资源时，来自被叫服务的资源策略仅授权 Amazon 服务主体，而不是配置呼叫服务的角色。例如，不附带任何条件地信任 CloudTrail 服务主体的 S3 存储桶可以从受信任的管理员配置的 Amazon Web Services 账户接收 CloudTrail 日志，但如果他们知道 S3 存储桶的名称，也可以从其 Amazon Web Services 账户中的未经授权角色那里接收 CloudTrail 日志。

当某个角色利用 Amazon 服务主体的信任来获得对他们本来不打算访问的资源的访问权限时，就会出现混淆的代理问题。

跨账户混淆代理人预防

下图阐明了跨账户混淆代理人问题。



此场景假定：

- Amazon1 是您的 Amazon Web Services 账户。
- Amazon1:ExampleRole 是您账户中的一个角色。该角色的信任策略通过将 Example Corp 的 Amazon 账户指定为可担任该角色的账户来信任 Example Corp。

将发生以下情况：

1. 在您开始使用 Example Corp 的服务时，您将向 Example Corp 提供 Amazon1:ExampleRole 的 ARN。
2. Example Corp 使用该角色 ARN 获取临时安全凭证以访问您的 Amazon Web Services 账户 中的资源。这样一来，您将信任 Example Corp 作为可代表您执行操作的“代理人”。
3. 另一个 Amazon 客户也开始使用 Example Corp 的服务，而且此客户还为 Example Corp 提供了 Amazon1:ExampleRole 的 ARN 以供其使用。另一个客户可能已了解或猜到已不是机密信息的 Amazon1:ExampleRole。
4. 当另一个客户要求 Example Corp 访问（它声称的）其账户中的 Amazon 资源时，Example Corp 使用 Amazon1:ExampleRole 访问您的账户中的资源。

这就是其他客户可对您的资源进行未授权访问的方式。由于此客户能够诱使 Example Corp 无意中操作您的资源，因此 Example Corp 现在是一个“混淆代理人”。

Example Corp 可以通过在角色信任策略中加入 `ExternalId` 条件检查来解决混淆的代理问题。Example Corp 为每个客户生成唯一的 `ExternalId` 值，并在其请求中使用该值来承担该角色。在 Example Corp 的客户中，`ExternalId` 值必须是独一无二的，并由 Example Corp 而不是其客户控制。这就是您从 Example Corp 获取该 ID 且不能自行提供该 ID 的原因。这可以防止 Example Corp 成为一个混淆代理人，以及授予对另一个账户的 Amazon 资源的访问权。

在我们的方案中，假设 Example Corp 为您提供的唯一标识符是 12345，而为另一个客户提供的标识符是 67890。这些标识符已针对此方案进行简化。通常，这些标识符为 GUID。假定这些标识符在 Example Corp 的客户之间是唯一的，它们将是用于外部 ID 的有意义的值。

Example Corp 将向您提供外部 ID 值 12345。然后，您必须将一个 `Condition` 元素添加到角色的信任策略，该策略要求 `sts:ExternalId` 值为 12345，如下所示：

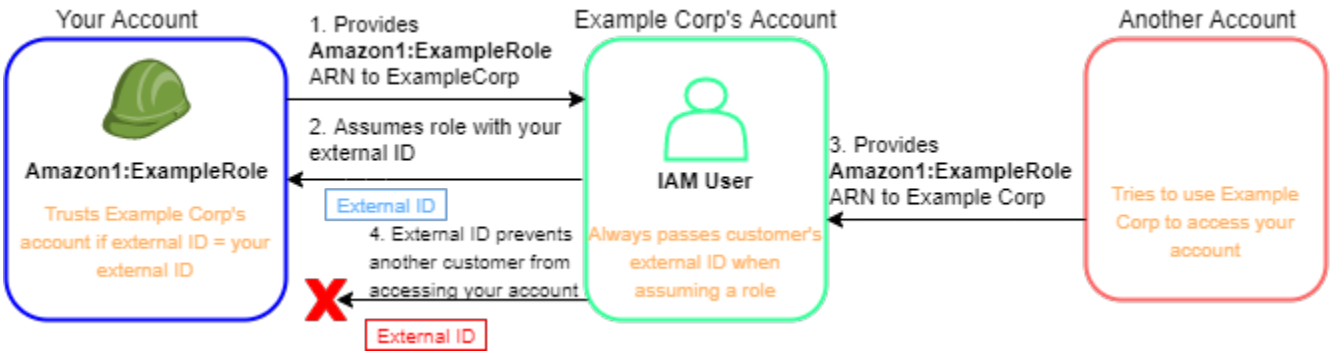
JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {
      "AWS": "Example Corp's Amazon Account ID"
    },
    "Action": "sts:AssumeRole",
```

```
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "12345"
      }
    }
  }
}
```

此策略中的 Condition 元素允许 Example Corp 仅在 AssumeRole API 调用包括外部 ID 值 12345 时代入该角色。Example Corp 确保，只要它代表客户担任角色，就会始终在 AssumeRole 调用中包括客户的外部 ID。即使另一个客户向 Example Corp 提供您的 ARN，也无法控制 Example Corp 包括在其发送给 Amazon 的请求中的外部 ID。这有助于防止未经授权的客户获取对您的资源的访问权限。

下图阐明了此过程。

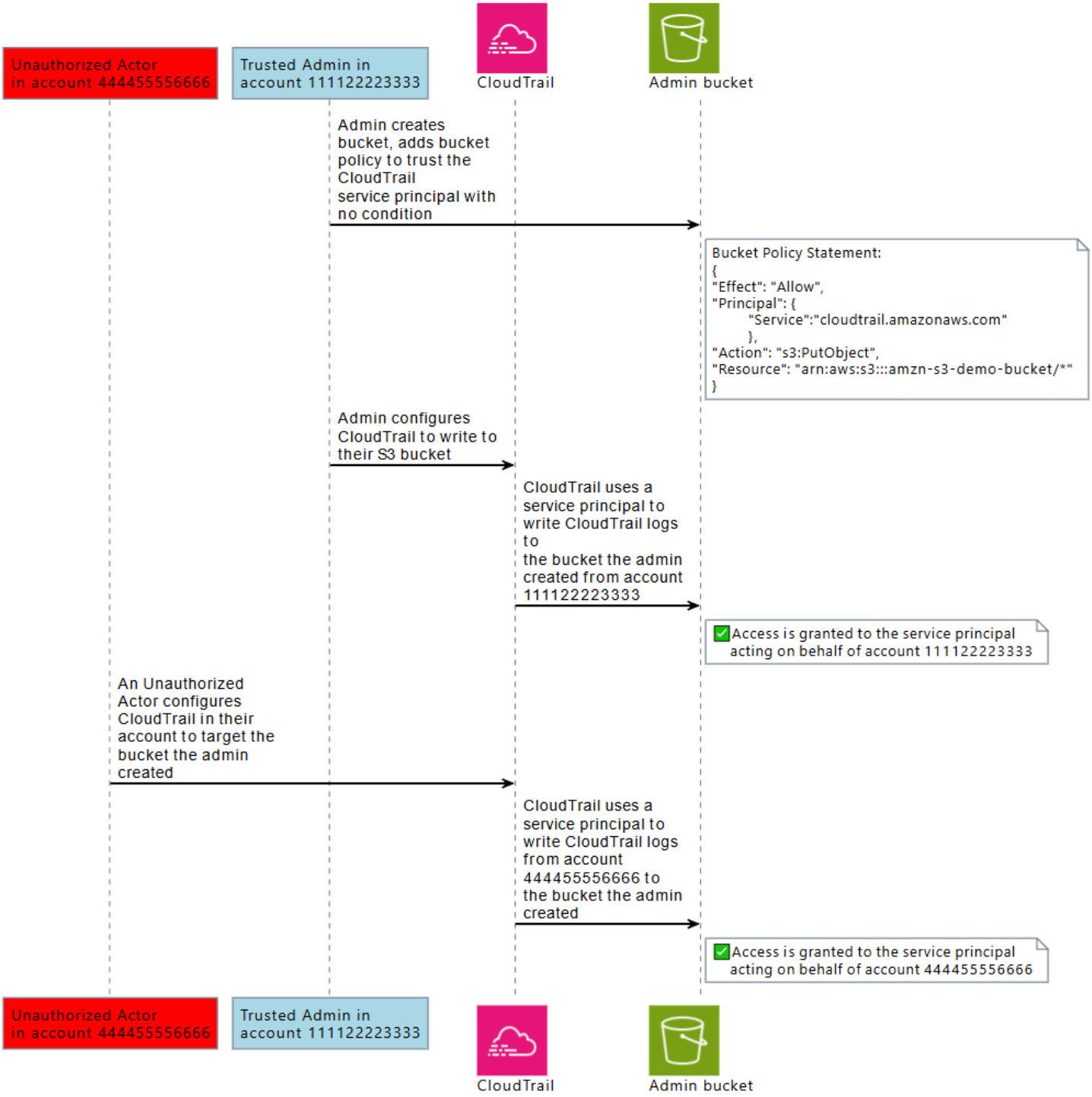


1. 和之前一样，在您开始使用 Example Corp 的服务时，您将向 Example Corp 提供 Amazon1:ExampleRole 的 ARN。
2. 在 Example Corp 使用该角色 ARN 来代入角色 Amazon1:ExampleRole 时，Example Corp 将在 AssumeRole API 调用中包含您的外部 ID（12345）。该外部 ID 与角色的信任策略匹配，因此 AssumeRole API 调用将成功，并且 Example Corp 将获取用于访问您的 Amazon Web Services 账户 中的资源的临时安全凭证。
3. 另一个 Amazon 客户也开始使用 Example Corp 的服务，而且如之前一样，此客户还为 Example Corp 提供了 Amazon1:ExampleRole 的 ARN 以供其使用。
4. 但这一次，在 Example Corp 尝试代入角色 Amazon1:ExampleRole 时，它提供了与另一个客户关联的外部 ID（67890）。另一个客户无法更改此外部 ID。Example Corp 这样做是因为另一个客户请求使用该角色，因此 67890 表示 Example Corp 正在其中操作的环境。由于您已将具有您自己的外部 ID（12345）的条件添加到 Amazon1:ExampleRole 的信任策略，因此 AssumeRole API 调用将失败。而且将阻止另一个客户对您账户中的资源进行未经授权的访问（由图中的红色“X”表示）。

外部 ID 有助于防止任何其他客户欺骗 Example Corp 不知不觉地访问您的资源。

防止跨服务混淆代理

下图使用 CloudTrail 和 Amazon S3 交互示例演示了跨服务的混淆代理问题，其中未经授权的角色将 CloudTrail 日志写入他们无权访问的 Amazon S3 存储桶。

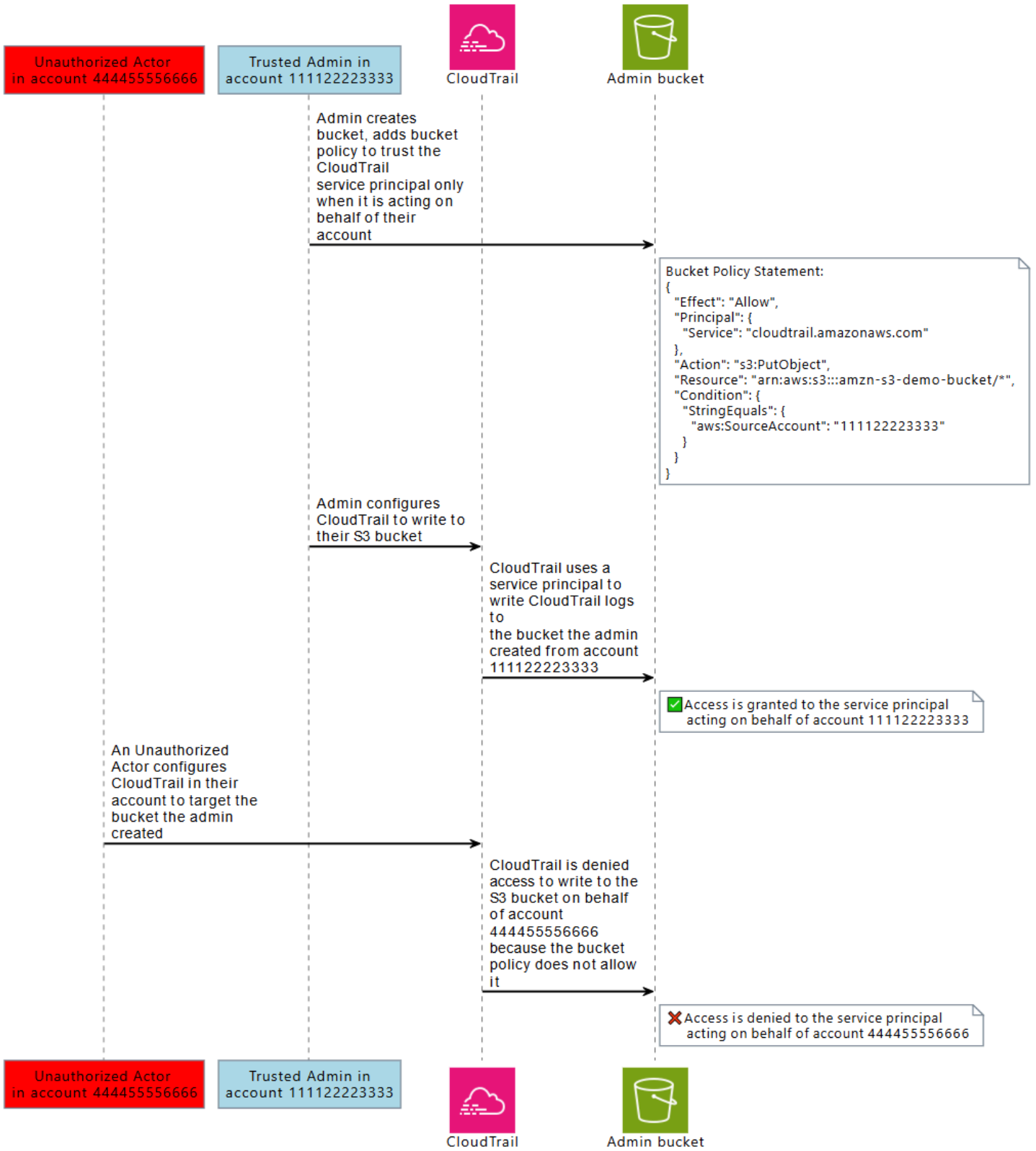


为了帮助防止未经授权的角色利用 Amazon 主体的信任来访问您的资源，Amazon 服务主体会提供有关他们所代表的 Amazon 资源、Amazon Web Services 账户和 Amazon 组织的信息。

此信息以全局条件键值的形式提供，这些键值可用于资源策略或 Amazon 服务主体发出的请求的资源控制策略。建议在您的资源策略中使用 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 或 aws:SourceOrgPaths，只要 Amazon 服务主体被授予访问您其中一个资源的权限。这些条件键使您能够在资源策略或资源控制策略中测试访问您资源的 Amazon 服务主体是否在代表您期望的 Amazon 资源、Amazon Web Services 账户或 Amazon Organizations 这样做。

- 使用 `aws:SourceArn` 允许 Amazon 服务主体代表特定资源（例如特定 Amazon CloudTrail 跟踪或 AppStream 实例集）访问您的资源。
- 使用 `aws:SourceAccount` 允许 Amazon 服务主体代表特定 Amazon Web Services 账户访问您的资源。
- 使用 `aws:SourceOrgID` 允许 Amazon 服务主体代表特定 Amazon Organizations 访问您的资源。
- 使用 `aws:SourceOrgPaths` 允许 Amazon 服务主体代表特定 Amazon Organizations 路径访问您的资源。

下图演示了跨服务的混淆代理场景，即使用 `aws:SourceAccount` 全局条件上下文键配置资源，而来自另一个账户的未经授权的角色试图访问他们本来不打算访问的 Amazon 资源。


Unauthorized Actor in account 444455556666
Trusted Admin in account 111122223333
CloudTrail
Admin bucket
Admin creates bucket, adds bucket policy to trust the CloudTrail service principal only when it is acting on behalf of their account
Bucket Policy Statement:
{
"Effect": "Allow",
"Principal": {
"Service": "cloudtrail.amazonaws.com"
},
"Action": "s3:PutObject",
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
"Condition": {
"StringEquals": {
"aws:SourceAccount": "111122223333"
}
}
}
Admin configures CloudTrail to write to their S3 bucket
CloudTrail uses a service principal to write CloudTrail logs to the bucket the admin created from account 111122223333
Access is granted to the service principal acting on behalf of account 111122223333
An Unauthorized Actor configures CloudTrail in their account to target the bucket the admin created
CloudTrail is denied access to write to the S3 bucket on behalf of account 444455556666 because the bucket policy does not allow it
Access is denied to the service principal acting on behalf of account 444455556666
Unauthorized Actor in account 444455556666
Trusted Admin in account 111122223333
CloudTrail
Admin bucket

在策略中使用 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 和 aws:SourceOrgPaths 全局条件键可帮助您确保服务主体代表您访问您的资源。建议每当向 Amazon 服务主体授予对其中一个资源的访问权限时，都使用这些条件键。

> **Note**
>
> 某些 Amazon Web Services 服务交互具有额外的控件，可帮助防止测试用户对资源访问的跨服务混淆代理问题。例如，当向 Amazon Web Services 服务发放 KMS 密钥授予时，Amazon KMS 会使用与资源关联的加密上下文以及密钥授予来帮助防止跨服务的混淆代理问题。有关帮助避免跨服务的混淆代理风险的更多信息，以及是否支持 aws:SourceArn、aws:SourceAccount、aws:SourceOrgID 和 aws:SourceOrgPaths，请参阅您使用是服务的文档。

使用基于资源的策略进行跨服务混淆代理保护

仅当服务主体代表 Amazon Web Services 账户 111122223333 行事时，以下示例策略授予服务主体 cloudtrail.amazonaws.com 访问 Amazon S3 存储桶，arn:aws:s3:::amzn-s3-demo-bucket1 的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CloudTrailAclCheck",
            "Effect": "Allow",
            "Principal": {"Service": "cloudtrail.amazonaws.com"},
            "Action": "s3:GetBucketAcl",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        },
        {
            "Sid": "AWSCloudTrailWrite",
            "Effect": "Allow",
```

```
            "Principal": {"Service": "cloudtrail.amazonaws.com"},
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/[optionalPrefix]/Logs/
myAccountID/*",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "111122223333"
                }
            }
        }
    ]
}
```

此示例存储桶策略仅在服务主体 appstream.amazonaws.com 通过使用 aws:SourceArn 指定实例集 ARN 来代表指定的 Amazon AppStream 实例集行事时，才授予服务主体访问 s3://amzn-s3-demo-bucket2 中的 powershell 脚本 examplefile.psh 的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "appstream.amazonaws.com"
                ]
            },
            "Action": "s3:GetObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket2/examplefile.psh",
            "Condition": {
                "ArnEquals": {
                    "aws:SourceArn": "arn:aws:appstream:us-
east-1:111122223333:fleet/ExampleFleetName"
                }
            }
        }
    ]
}
```

使用资源控制策略进行跨服务混淆代理保护

您可以使用资源控制策略（RCP）将跨服务的混淆代理控制应用于支持的 Amazon Web Services 服务的资源。RCP 允许您对资源集中应用跨服务的混淆代理控制。您可以将条件键（如 `aws:SourceOrgId` 和 `aws:SourceOrgPaths`）与附加到您的 Amazon Organizations 的 RCP、组织单位（OU）或组织中的 Amazon Web Services 账户一起使用，而无需在基于资源的特定策略中添加声明。有关 RCP 和支持的服务更多信息，请参阅《Amazon Organizations 用户指南》中的资源控制策略（RCP）。

以下示例 RCP 拒绝 Amazon 服务主体在 `aws:SourceOrgID` 不等于 o-ExampleOrg 时访问您的成员账户中的 Amazon S3 存储桶。S3 存储桶的基于资源的策略中必须存在相应的允许，以允许 `SourceOrgID` 等于 o-ExampleOrg 的 Amazon Web Services 服务主体。

此策略仅对存在 `aws:SourceAccount` 密钥（`"Null": {"aws:SourceAccount": "false"}`）的服务主体（`"Bool": {"aws:PrincipalIsAWSService": "true"}`）进行的请求应用控制，因此不需要使用此条件键的服务集成和主体的调用不会受到影响。如果 `aws:SourceAccount` 条件键存在于请求上下文中，则 Null 条件将评估为 true，从而强制执行 `aws:SourceOrgID`。我们在 Null 条件运算符中使用 `aws:SourceAccount` 而不是 `aws:SourceOrgID`，因此，如果请求来自不属于组织的账户，则控制仍然适用。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "RCPEnforceConfusedDeputyProtectionForS3",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceOrgID": "o-ExampleOrg"
        },
        "Null": {
          "aws:SourceAccount": "false"
        },
```

```
        "Bool": {
          "aws:PrincipalIsAWSService": "true"
        }
      }
    }
  ]
}
```

IAM 角色的常见场景

与大多数 Amazon 功能一样，您通常可按照两种方式来使用角色：在 IAM 控制台中以交互方式使用角色，或通过 Amazon CLI、Tools for Windows PowerShell 或 API 以编程方式使用角色。

- 您账户中的使用 IAM 控制台的 IAM 用户可切换为某个角色，以临时使用该控制台中的角色的权限。该用户放弃自己的原始权限，采用分配给该角色的权限。用户退出角色时，将恢复其原始权限。
- 应用程序或 Amazon 提供的服务（如 Amazon EC2）可以通过请求供角色用来向 Amazon 进行编程请求的临时安全凭证来代入角色。您可以按这种方式使用角色，以便不必为需要访问资源的每个实体共享或维护长期安全凭证 (例如，通过创建 IAM 用户)。

> **Note**
>
> 本指南以可互换方式使用了切换到角色和代入角色这两个短语。

最简单的角色使用方式是向 IAM 用户授予权限，以便切换为您在自己或其他 Amazon Web Services 账户 中创建的角色。他们可以使用 IAM 控制台轻松地切换角色以使用通常您不希望他们拥有的权限，然后退出角色以交出这些权限。这样有助于防止对敏感资源进行意外 访问或修改。

若要对角色进行更复杂的使用 (如向应用程序和服务或联合身份外部用户授予访问权限)，可以调用 `AssumeRole` API。该 API 调用返回应用程序可在后续 API 调用中使用的一组临时凭证。使用临时凭证执行的操作只拥有相关联的角色所授予的权限。应用程序不必像控制台中的用户那样“退出”角色；而是应用程序直接停止使用临时凭证并使用原始凭证继续调用。

联合身份用户使用身份提供程序 (IdP) 提供的凭证登录。Amazon 随后向可信 IdP 提供临时证书以传递给用户，以便包含在后续 Amazon 资源请求中。这些证书提供向分配的角色授予的权限。

本部分提供了以下方案的概述：

- 为您拥有的一个 Amazon Web Services 账户 中的 IAM 用户提供对您拥有的其他账户中的资源的访问权限
- 提供对非 Amazon 工作负载的访问权限
- 为第三方拥有的 Amazon Web Services 账户 中的 IAM 用户提供访问权限
- 为 Amazon 提供的服务提供对 Amazon 资源的访问权限
- 为经过外部身份验证的用户（联合身份验证）提供访问权限

在您拥有的其他 Amazon Web Services 账户 中 IAM 用户的访问权限

您可以向 IAM 用户授予权限，以便切换至您 Amazon Web Services 账户 中的角色，或切换至您拥有的其他 Amazon Web Services 账户 中定义的角色。

> **Note**
>
> 如果要授予对您未拥有或无法控制的账户的访问权限，请参阅本主题后面的访问第三方拥有的 Amazon Web Services 账户。

假设您拥有一个企业关键型 Amazon EC2 实例。您可以创建具有这些权限的角色，而不是直接向用户授予终止实例的权限。然后，允许管理员在需要终止实例时切换为该角色。这样，可向实例添加以下几层保护：

- 您必须向用户显式授予担任该角色的权限。
- 用户必须使用 Amazon Web Services 管理控制台主动切换为该角色，或使用 Amazon CLI 或 Amazon API 担任角色。
- 您可以向该角色添加多重身份验证 (MFA) 保护，以便只有登录 MFA 设备的用户才能担任该角色。要了解如何配置角色以使担任角色的用户必须先使用多重身份验证 (MFA) 进行身份验证，请参阅使用 MFA 保护 API 访问。

我们建议使用该方法以实施最低权限原则。这意味着，仅限特定任务需要时，才能使用提升的权限。借助角色，您可以帮助防止对敏感环境进行意外更改，如果您将它们与审核合并以帮助确保仅在需要时才使用角色，这尤其适用。

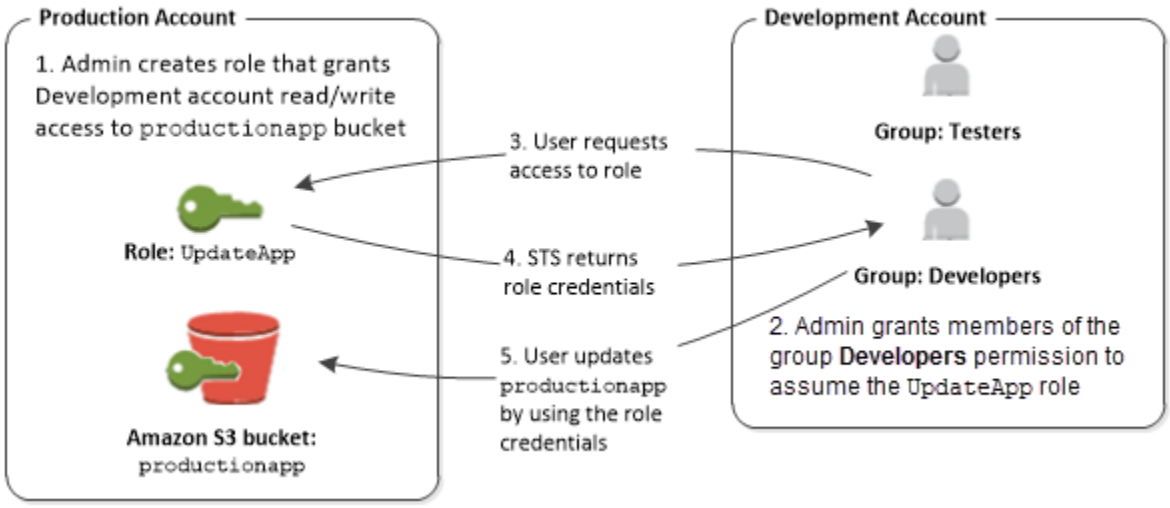
在您出于此目的创建角色时，可在该角色的信任策略的 `Principal` 元素中按 ID 指定其用户需要访问权限的账户。随后可以向这些其他账户中的特定用户授予切换到角色的权限。要了解您信任区域

之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

一个账户中的用户可以切换为相同或不同账户中的角色。使用角色过程中，用户只能执行角色允许的操作并且只能访问角色允许的资源；其原始用户权限处于暂停状态。用户退出角色时，恢复原始用户权限。

使用不同的开发和生产账户的示例方案

假设您的组织拥有多个 Amazon Web Services 账户 以将开发环境与生产环境隔离。开发账户中的用户有时可能需要访问生产账户中的资源。例如，在促进从开发环境到生产环境的更新时，您可能需要进行跨账户访问。尽管您可以为在两个账户中工作的用户创建单独的身份 (和密码)，多个账户的凭证管理还是会为身份管理带来难题。在以下图表中，所有用户都通过开发账户进行管理，但一些开发人员需要对生产账户进行有限访问。开发账户有两个组：测试人员和开发人员，每个组有其自身的策略。



1. 在生产账户中，管理员使用 IAM 在该账户中创建 UpdateApp 角色。在角色中，管理员定义信任策略，该策略将开发账户指定为 Principal，这意味着开发账户中的授权用户可以使用 UpdateApp 角色。管理员还为角色定义权限策略，以指定对名为 productionapp 的 Amazon S3 存储桶的读取和写入权限。

   然后，管理员相应信息与需要担任角色的所有人共享。该信息是角色的账号和名称（对于 Amazon 控制台用户）或 Amazon 资源名称 (ARN)（用于 Amazon CLI 或 Amazon API 访问）。角色 ARN 类似于 arn:aws:iam::123456789012:role/UpdateApp，其中角色名为 UpdateApp，角色采用账号 123456789012 进行创建。

> **Note**
>
> 管理员可以选择配置角色，以便担任角色的用户必须先使用多重身份验证 (MFA) 进行身份验证。有关更多信息，请参阅 使用 MFA 保护 API 访问。

2. 在开发账户中，管理员向开发人员组的成员授予切换为角色的权限。执行此操作的方法是向开发人员组授予针对 `UpdateApp` 角色调用 Amazon Security Token Service (Amazon STS) `AssumeRole` API 的权限。开发账户中开发人员组的所有 IAM 用户现在都可以切换为生产账户中的 `UpdateApp` 角色。不在开发人员组中的其他用户无权切换为该角色，因此无法访问生产账户中的 S3 存储桶。
3. 用户请求切换为角色：
   - Amazon 控制台：用户选择导航栏上的账户名并选择 Switch Role (切换角色)。用户指定账户 ID (或别名) 和角色名称。或者，用户可以单击管理员在电子邮件中发送的链接。通过该链接，用户可以转到已填写详细信息的 Switch Role 页面。
   - Amazon API/Amazon CLI：开发账户中开发人员组的用户调用 `AssumeRole` 函数以获取 `UpdateApp` 角色的凭证。用户将 `UpdateApp` 角色的 ARN 指定为调用的一部分。如果测试人员组中的用户发出相同请求，请求将失败，因为测试人员没有针对 `UpdateApp` 角色 ARN 调用 `AssumeRole` 的权限。
4. Amazon STS 返回临时凭证：
   - Amazon 控制台：Amazon STS 使用角色的信任策略来验证请求，以确保请求来自受信任实体（即开发账户）。验证完成后，Amazon STS 向 Amazon 控制台返回临时安全凭证。
   - API/CLI：Amazon STS 根据角色的信任策略来验证请求，以确保请求来自受信任实体 (即开发账户)。验证完成后，Amazon STS 向应用程序返回临时安全凭证。
5. 临时凭证允许访问 Amazon 资源：
   - Amazon 控制台：Amazon 控制台在所有后续控制台操作中代表用户使用临时凭证，在本例中是用于读取和写入 `productionapp` 存储桶。该控制台无法访问生产账户中的任何其他资源。用户退出角色时，用户的权限恢复为切换为角色之前所拥有的原始权限。
   - API/CLI：应用程序使用临时安全凭证更新 `productionapp` 存储桶。应用程序只能使用临时安全凭证读取和写入 `productionapp` 存储桶，无法访问生产账户的任何其他资源。应用程序不必退出角色，只需在后续 API 调用中停止使用临时凭证并使用原始凭证。

其他资源

有关更多信息，请参阅下列内容：

- IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限

非 Amazon 工作负载的访问权限

IAM 角色是 Amazon Identity and Access Management（IAM）中分配权限的对象。当您担任的角色使用 IAM 身份或来自 Amazon 外部的身份时，它会为您提供角色会话的临时安全凭证。您的工作负载可能在数据中心或其他 Amazon 外部必须访问您 Amazon 资源的基础设施中运行。除了创建、分发和管理长期访问密钥之外，您可以使用 Amazon Identity and Access Management Roles Anywhere（IAM Roles Anywhere）来验证您的非 Amazon 工作负载。IAM Roles Anywhere 使用您的证书颁发机构（CA）颁发的 X.509 证书对身份进行验证，并安全地使用 IAM 角色提供的临时证书提供对 Amazon Web Services 服务 的访问权限。

使用 IAM Roles Anywhere

1. 设置使用 Amazon 私有证书颁发机构 的 CA 或者使用来自您自己的 PKI 基础设施的 CA。
2. 设置 CA 后，您可以在 IAM Roles Anywhere 中创建一个称为信任锚的对象。此锚点在 IAM Roles Anywhere 和您的 CA 之间建立信任以进行身份验证。
3. 然后，您可以配置现有的 IAM 角色，或者创建信任 IAM Roles Anywhere 服务的新角色。
4. 使用信任锚通过 IAM Roles Anywhere 对非 Amazon 工作负载进行身份验证。Amazon 将非 Amazon 工作负载临时凭证授予有权访问您的 Amazon 资源的 IAM 角色。

其他资源

以下资源可以帮助您了解有关提供非 Amazon 工作负载访问权限的更多信息。

-
- 要了解如何设置 IAM Roles Anywhere 的公有密钥基础设施（PKI），请参阅Amazon安全博客中的 IAM Roles Anywhere with an external certificate authority。

访问第三方拥有的 Amazon Web Services 账户

当第三方请求访问您的组织的 Amazon 资源时，您可以使用角色向他们委派访问权限。例如，第三方可能某种服务来管理您的 Amazon 资源。您可以通过 IAM 角色授予第三方访问您的 Amazon 资源的权限，而无需共享您的 Amazon 安全凭证。而第三方则可以通过代入您在 Amazon Web Services 账户中创建的角色来访问您的 Amazon 资源。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

为了创建他们可以代入的角色，第三方必须为您提供以下信息：

- 第三方的 Amazon Web Services 账户 ID。为角色定义信任策略时，可将其 Amazon Web Services 账户 ID 指定为主体。
- 与角色唯一关联的外部 ID。外部 ID 可以是仅您和第三方已知的任何标识符。例如，您可以使用您与该第三方之间的发票 ID，但不要使用能被猜到的内容，如第三方的名称或电话号码。为角色定义信任策略时，必须指定该 ID。第三方在代入角色时必须提供该 ID。
- 第三方使用您的 Amazon 资源所需的权限。在定义角色的权限策略时，您必须指定这些权限。这个策略定义了他们可以执行哪些操作以及可以访问哪些资源。

创建完角色后，您必须向第三方提供该角色的 Amazon Resource Name (ARN)。他们需要使用您的角色的 ARN 来代入该角色。

> **Important**
>
> 当您授予第三方访问 Amazon 资源的权限时，他们可以访问您在该策略中指定的任何资源。系统会为您发送他们使用资源的记录。请确保适当限制他们对资源的使用。

用于第三方访问的外部 ID

外部 ID 允许正担任该角色的用户断言其运行的环境。它还为账户所有者提供一种方法来允许仅在特定情况下担任该角色。外部 ID 的主要功能是解决并防止 混淆代理人问题。

> **Important**
>
> Amazon 不会将该外部 ID 作为秘密信息。您在 Amazon 中创建访问密钥对或密码这类秘密信息后，您无法再次查看它们。有权查看角色的任何人都可以看到该角色的外部 ID。

我何时应使用外部 ID？

在以下情况下使用外部 ID：

- 您是 Amazon Web Services 账户 所有者并且已为将访问其他 Amazon Web Services 账户 以及您的账户的第三方配置角色。您应要求第三方提供其在担任您的角色时包含的外部 ID。然后，在您角色的信任策略中检查该外部 ID。这样做可确保外部方仅在代表您执行操作时才能担任您的角色。
- 在前述情况下，您代表不同客户 (如 Example Corp) 担任角色。您应该为每个客户分配一个唯一的外部 ID 并指导他们将该外部 ID 添加到其角色的信任策略。然后，您必须确保在担任角色的请求中始终包含正确的外部 ID。

您可能已为您的每个客户提供一个唯一标识符，而且此唯一 ID 足以用作外部 ID。该外部 ID 不是您要明确创建或分别跟踪所需的特殊值 (仅用于此目的)。

您应始终在您的 `AssumeRole` API 调用中指定外部 ID。此外，在客户为您提供角色 ARN 时，请测试是否能在带有/不带正确外部 ID 的情况下担任该角色。如果可在没有正确外部 ID 的情况下担任角色，则不要在您的系统中存储该客户的角色 ARN。等待该客户将角色信任策略更新为要求提供正确的外部 ID。这样一来，您帮助您的客户执行了正确的操作，并帮助您和客户避免了混淆代理人问题。

使用外部 ID 的示例场景

例如，假设您决定聘请一家名为 Example Corp 的第三方公司来监控您的 Amazon Web Services 账户并帮助优化成本。为跟踪您的日常开支，Example Corp 需要访问您的 Amazon 资源。Example Corp 也可监控其他客户的很多其他 Amazon 账户。

请勿向 Example Corp 提供对您的 Amazon 账户中的 IAM 用户及其长期凭证的访问权限。请使用 IAM 角色 角色及其临时安全凭证。IAM 角色提供了一种机制，可允许第三方访问您的 Amazon 资源而无需共享长期凭证（例如，IAM 用户的访问密钥）。

您可使用 IAM 角色在您的 Amazon Web Services 账户 和 Example Corp 账户之间建立信任关系。建立这种关系后，Example Corp 账户的成员可以调用 Amazon Security Token Service AssumeRole API 以获取临时安全凭证。然后，Example Corp 成员可以使用这些凭证访问您的账户中的 Amazon 资源。

> **Note**
>
> 有关可调用来获取临时安全凭证的 AssumeRole 和其他 Amazon API 操作的更多信息，请参阅比较 Amazon STS 凭证。

以下是此方案的更多详细信息：

1. 您聘请了 Example Corp，此公司将为您创建唯一客户标识符。他们为您提供此唯一客户 ID 及其 Amazon Web Services 账户 账号。您需要此信息来在下一步中创建 IAM 角色。

   > **Note**
   >
   > Example Corp 可使用其想用于 ExternalId 的任何字符串值，只要该值对于每个客户来说都是唯一的。该值可以是客户账号，甚至可以是一个随机字符串，只要没有两个客户拥有相同

的值即可。该值不是“机密的”。Example Corp 必须向每个客户提供 ExternalId 值。关键的一点是，该值必须由 Example Corp 生成，而不是其客户生成，从而确保每个外部 ID 是唯一的。

2. 您登录到 Amazon 并创建 IAM 角色，该角色可向 Example Corp 提供对您的资源的访问权。与任何 IAM 角色类似，该角色具有两个策略：权限策略和信任策略。该角色的信任策略规定谁可担任该角色。在我们的示例方案中，该策略将 Example Corp 的 Amazon Web Services 账户 账号指定为 `Principal`。这允许来自此账户的身份担任该角色。此外，将 `Condition` 元素添加到信任策略。此 `Condition` 测试 `ExternalId` 上下文密钥，以确保它与 Example Corp 的唯一客户 ID 一致。例如：

```
   "Principal": {"AWS": "Example Corp's Amazon Web Services ## ID"},
   "Condition": {"StringEquals": {"sts:ExternalId": "Unique ID Assigned by Example
Corp"}}
```

3. 该角色的权限策略指定该角色允许某个人执行哪些操作。例如，您可以规定该角色允许某人仅管理您的 Amazon EC2 和 Amazon RDS 资源，但不能管理您的 IAM 用户或组。在我们的示例方案中，您使用权限策略为 Example Corp 授予您的账户中的所有资源的只读访问权限。
4. 创建完角色后，您向 Example Corp 提供该角色的 Amazon Resource Name (ARN)。
5. 当 Example Corp 需要访问您的 Amazon 资源时，该公司的某人可调用 Amazon `sts:AssumeRole` API。该调用包括要担任的角色的 ARN 以及与其客户 ID 对应的 ExternalId 参数。

如果发出请求的人使用的是 Example Corp 的 Amazon Web Services 账户，并且角色 ARN 和外部 ID 是正确的，则请求成功。然后，该请求提供临时安全凭证，Example Corp 可使用这些凭证访问您的角色允许访问的 Amazon 资源。

换句话说，当角色策略包括外部 ID 时，任何需要担任该角色的人都必须是该角色中的主体，还必须包括正确的外部 ID。

外部 ID 的要点

- 在您支持多个客户拥有不同 Amazon 账户的多租户环境中，我们建议每个 Amazon Web Services 账户 使用一个外部 ID。此 ID 应该是第三方生成的随机字符串。
- 要在代入角色时需要提供外部 ID 的第三方，请使用您选择的外部 ID 来更新角色的信任策略。
- 要在代入角色时提供外部 ID，请使用 Amazon CLI 或 Amazon API 来代入该角色。有关更多信息，请参阅 STS AssumeRole API 操作或 STS assume-role CLI 操作。

- `ExternalId` 值的长度必须最少为 2 个字符，最多为 1224 个字符。该值必须是字母数字，没有空格。它还可以包含以下符号：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、@ 符号、冒号 (:)、正斜杠 (/) 和连字符 (-)。

其他资源

以下资源可帮助您了解有关向第三方拥有的 Amazon Web Services 账户 提供访问权限的更多信息。

- 要了解如何允许其他人在您的 Amazon Web Services 账户 中执行操作，请参阅 使用自定义信任策略创建角色。
- 要了解如何授予切换到角色的权限，请参阅 向用户授予切换角色的权限
- 要了解如何创建并向受信任的用户提供临时安全凭证，请参阅 临时安全凭证的权限。

访问 Amazon 服务

很多 Amazon 服务要求您利用角色控制该服务可以访问的内容。由一项服务担任、代表您执行操作的角色称为服务角色。当某角色为某项服务提供专门用途时，可以将其归类为服务相关角色。请参阅每种服务的 Amazon 文档，以了解它是否使用角色以及如何分配角色以供服务使用。

有关创建角色以向 Amazon 提供的服务委派访问权限的详细信息，请参阅创建向 Amazon 服务委派权限的角色。

访问经过外部身份验证的用户（身份联合验证）

您的用户可能已具有 Amazon 外部的身份，如在公司目录中。如果这些用户需要使用 Amazon 资源（或使用访问这些资源的应用程序），则这些用户也需要 Amazon 安全凭证。您可以使用 IAM 角色为身份通过您的企业或第三方身份提供者 (IdP) 进行联合证明的用户指定权限。

> **Note**
>
> 作为安全最佳实践，我们建议您使用身份联合验证在 IAM Identity Center 中管理用户访问权限，而不是创建 IAM 用户。要了解需要使用 IAM 用户的特定情况，请参阅 何时创建 IAM 用户（而非角色）。

通过 Amazon Cognito 联合移动或基于 Web 的应用程序的用户

如果创建访问 Amazon 资源的移动或基于 Web 的应用程序，则应用程序需要安全凭证才能向 Amazon 进行编程请求。对于大多数移动应用程序情况，我们建议您使用 Amazon Cognito。您可以将此服务与

Amazon Mobile SDK for iOS 和 Amazon Mobile SDK for Android and Fire OS 搭配使用，为用户创建唯一的身份，并对其进行身份验证，以使其安全地访问您的 Amazon 资源。Amazon Cognito 支持与下一个部分中所列的那些提供商相同的身份提供者，而且还支持已经过开发人员验证的身份和未经身份验证的（来宾）访问。Amazon Cognito 还提供用于同步用户数据的 API 操作，因此，无论用户在设备间怎样转移，其数据总能得以保留。有关更多信息，请参阅 用于移动应用程序的 Amazon Cognito。

使用公共身份服务提供商或 OpenID Connect 联合身份用户

在移动和基于 Web 的场景下尽可能使用 Amazon Cognito。Amazon Cognito 为您提供公共身份提供者服务的大部分幕后工作。它与相同的第三方服务协作，也支持匿名登录。不过，对于更高级的方案，您可以直接使用第三方服务，例如 Login with Amazon、Facebook、Google 或任何与 OpenID Connect (OIDC) 兼容的 IdP。有关通过这些服务之一使用 OIDC 联合身份验证的更多信息，请参阅 OIDC 联合身份验证。

通过 SAML 2.0 联合身份用户

如果您的组织已使用支持 SAML 2.0 (安全断言标记语言 2.0) 的身份提供者软件包，您可以在作为身份提供者 (IdP) 的您的组织和作为服务提供商的 Amazon 之间建立信任。随后可以使用 SAML 为您的用户提供对 Amazon Web Services 管理控制台的联合单点登录 (SSO)，或是用于调用 Amazon API 操作的联合访问。例如，如果您的公司使用 Microsoft Active Directory 和 Active Directory 联合身份验证服务，则您可以使用 SAML 2.0 进行联合。有关使用 SAML 2.0 对用户进行联合身份验证的更多信息，请参阅SAML 2.0 联合身份验证。

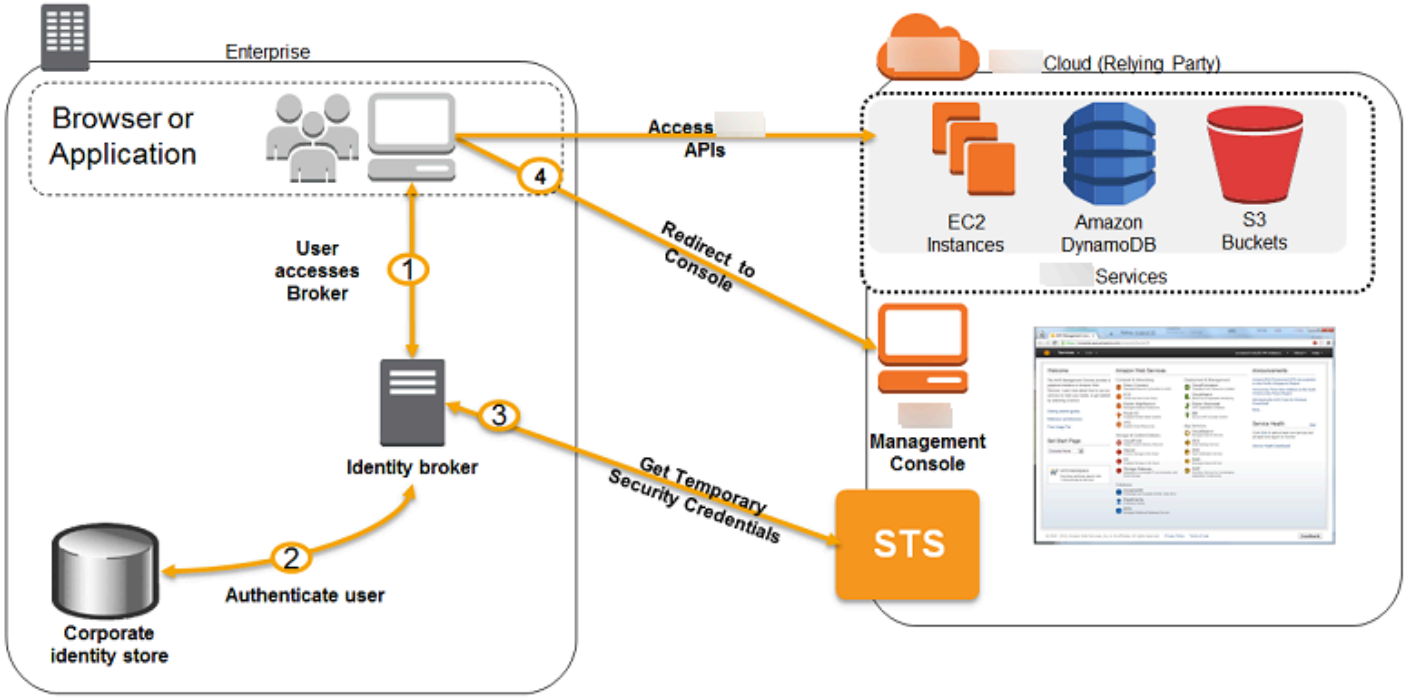
通过创建自定义身份凭证代理程序来联合用户

如果您的身份存储与 SAML 2.0 不兼容，则可构建自定义身份凭证代理程序来执行类似功能。代理应用程序将对用户进行身份验证，从 Amazon 为用户请求临时安全凭证，然后将凭证提供给用户以便访问 Amazon 资源。

例如，Example Corp. 有很多员工需要运行访问公司 Amazon 资源的内部应用程序。这些员工在公司的身份与验证系统中已有身份，并且 Example Corp. 不希望为每位公司员工都创建一个单独的 IAM 用户。

Bob 是 Example Corp. 的一名开发人员。为使 Example Corp. 的内部应用程序可访问公司的 Amazon 资源，Bob 开发了一个自定义身份凭证代理程序。该应用程序确认员工登录到现有的 Example Corp. 身份与验证系统，而该系统可能使用 LDAP、Active Directory 或其他系统。然后，身份凭证代理程序获取员工的临时安全凭证。此场景与以前的某个场景（一个使用自定义身份验证系统的移动应用程序）类似，但需要访问 Amazon 资源的应用程序全都运行在企业网络中，并且公司现有一个身份验证系统。

为了获取临时安全凭证，该身份凭证代理程序调用 AssumeRole 或 GetFederationToken 以获取临时安全凭证，具体取决于 Bob 要如何管理用户的策略以及临时凭证应何时到期。(有关这些 API 操作之间的差异的更多信息，请参阅IAM 临时安全凭证和临时安全凭证的权限。) 该调用返回由 Amazon 访问密钥 ID、秘密访问密钥和会话令牌组成的临时安全凭证。该身份凭证代理程序向内部公司应用程序提供这些临时安全凭证。然后，该应用程序可使用这些临时凭证直接调用 Amazon。该应用程序将证书缓存至其到期，然后请求新的一组临时证书。下图举例说明此场景。



此方案有以下属性：

- 该身份凭证代理程序具有访问 IAM 的令牌服务 (STS) API 以创建临时安全凭证的权限。
- 该身份凭证代理程序可确认员工在现有的身份验证系统中经过身份验证。
- 用户可获得一个临时 URL，通过它可访问 Amazon 管理控制台 (称为单点登录)。

有关创建临时安全凭证的信息，请参阅比较 Amazon STS 凭证。有关获取对 Amazon 管理控制台的访问权限的 SAML 联合主体的更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。

IAM 角色创建

要创建角色，可以使用 Amazon Web Services 管理控制台、Amazon CLI、Tools for Windows PowerShell 或 IAM API。

如果使用的是 Amazon Web Services 管理控制台，则可使用向导来完成创建角色的步骤。向导的步骤可能稍有不同，具体取决于您是为 Amazon 服务、Amazon Web Services 账户 还是 SAML 或 OIDC 联合主体创建角色。

IAM 用户的角色

创建此角色，以在您的 Amazon Web Services 账户 内或向您拥有的其他 Amazon Web Services 账户中定义的角色委派权限。一个账户中的用户可以切换为相同或不同账户中的角色。使用角色过程中，用户只能执行角色允许的操作并且只能访问角色允许的资源；其原始用户权限处于暂停状态。用户退出角色时，恢复原始用户权限。

有关更多信息，请参阅 创建角色，向 IAM 用户授予权限。

有关创建跨账户访问角色的更多信息，请参阅 使用自定义信任策略创建角色。

Amazon 服务的角色

创建此角色以将权限委派给可以代表您执行操作的服务。您传递给服务的服务角色必须具有 IAM 策略，该策略具有允许服务执行与该服务关联的操作的权限。每项 Amazon 服务都需要不同的权限。

有关创建服务角色的更多信息，请参阅 创建向 Amazon 服务委派权限的角色。

有关创建服务相关角色的更多信息，请参阅 创建服务相关角色。

身份联合验证的角色

创建此角色可将权限委派给已在 Amazon 外部拥有身份的用户。使用身份提供商时，您不必创建自定义登录代码或管理自己的用户身份。您的外部用户通过 IdP 登录，您可以向这些外部身份授予使用您的账户中的 Amazon 资源的权限。身份提供商可帮助您确保 Amazon 账户的安全，因为您不必在应用程序中分配或嵌入长期安全凭证（如访问密钥）。

有关更多信息，请参阅 为第三方身份提供者创建角色。

创建角色，向 IAM 用户授予权限

您可以使用 IAM 角色提供对 Amazon 资源的访问权限。利用 IAM 角色，您可以在您的信任账户和其他 Amazon 受信任账户之间建立信任关系。信任账户拥有要访问的资源，受信任账户包含需要资源访问权限的用户。但是，其他账户可以拥有您的账户中的资源。例如，信任账户可能允许受信任账户创建新资源，例如在 Amazon S3 存储桶中创建新对象。在这种情况下，创建资源的账户拥有资源并控制谁可以访问该资源。

创建信任关系后，来自受信任账户的 IAM 用户或应用程序可以使用 Amazon Security Token Service (Amazon STS) `AssumeRole` API 操作。此操作提供临时安全凭证，可用于访问您账户中的 Amazon 资源。

账户可由您进行控制，或者具有用户的账户可由第三方进行控制。如果其他具有用户的账户是您无法控制的 Amazon Web Services 账户，则可使用 `externalId` 属性。外部 ID 可以是您和第三方账户管理员之间达成一致的任何字词或数字。此选项会在信任策略中自动添加一个条件，即仅当请求包括正确的 `sts:ExternalID` 时，该用户才担任该角色。有关更多信息，请参阅 访问第三方拥有的 Amazon Web Services 账户。

有关如何使用角色委派权限的信息，请参阅角色术语和概念。有关使用服务角色以允许服务访问账户中的资源的信息，请参阅创建向 Amazon 服务委派权限的角色。

创建 IAM 角色（控制台）

您可以使用 Amazon Web Services 管理控制台 创建 IAM 用户可担任的角色。例如，假设贵组织拥有多个 Amazon Web Services 账户 以便将开发环境与生产环境隔离。有关创建角色（该角色允许开发账户中的用户访问生产账户中的资源）的概述信息，请参阅 使用不同的开发和生产账户的示例方案。

最小权限

要执行下列步骤，您必须至少具有以下 IAM 权限：

- `access-analyzer:ValidatePolicy`
- `iam:AttachRolePolicy`
- `iam:CreatePolicy`
- `iam:CreateRole`
- `iam:GetAccountSummary`
- `iam:GetPolicy`
- `iam:GetPolicyVersion`
- `iam:GetRole`
- `iam:ListAccountAliases`
- `iam:ListAttachedRolePolicies`
- `iam:ListOpenIDConnectProviders`
- `iam:ListPolicies`
- `iam:ListRolePolicies`

- `iam:ListRoles`
- `iam:ListRoleTags`
- `iam:ListSAMLProviders`

Console

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Amazon Web Services 账户 角色类型。
4. 要为您的账户创建角色，请选择 This account（此账户）。要为其他账户创建角色，请选择其他 Amazon Web Services 账户，然后输入要向其授予资源访问权限的账户 ID。

   指定账户的管理员可向该账户中的任何 IAM 用户授予担任该角色的权限。为此，管理员需要将策略附加到用户或组来授予 `sts:AssumeRole` 操作的权限。该策略必须指定角色的 ARN 作为 `Resource`。
5. 如果您向不受您控制的账户的用户授予权限，并且用户将以编程方式代入此角色，请选择 Require external ID（需要外部 ID）。外部 ID 可以是您和第三方账户管理员之间达成一致的任意字词或数字。此选项会在信任策略中自动添加一个条件，即仅当请求包括正确的 `sts:ExternalID` 时，该用户才担任该角色。有关更多信息，请参阅 访问第三方拥有的 Amazon Web Services 账户。

   > **Important**
   >
   > 选择此选项将仅允许通过 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 访问该角色。这是因为，您不能使用 Amazon 控制台切换到其信任策略中有 `externalId` 条件的角色。但您可以通过使用相关开发工具包编写脚本或应用程序，以编程方式创建此类访问。有关更多信息和示例脚本，请参阅 Amazon 安全博客中的如何启用对 Amazon Web Services 管理控制台 的跨账户存取。
6. 如果需要将该角色限制为使用多重身份验证 (MFA) 进行登录的用户，请选择 Require MFA。这将向角色的信任策略中添加用于检查 MFA 登录的条件。需要担任角色的用户必须使用配置的 MFA 设备中的临时一次性密码进行登录。没有经过 MFA 身份验证的用户无法担任该角色。有关 MFA 的更多信息，请参阅IAM 中的 Amazon 多重身份验证
7. 选择下一步。

8. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或选择创建策略以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅创建 IAM 策略。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望担任角色的任何人具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
9. （可选）设置权限边界。这是一项高级功能。

   打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。选择要用于权限边界的策略。
10. 选择下一步。
11. 对于角色名称，请为您的角色输入一个名称。角色名称在您的 Amazon Web Services 账户内必须是唯一的。在策略中使用角色名称或将其作为 ARN 的一部分时，角色名称区分大小写。在控制台中向客户显示角色名称时（例如在登录过程中），角色名称不区分大小写。由于多个实体可能引用该角色，因此，角色创建完毕后，您将无法编辑角色名称。
12. （可选）对于 Description（描述），输入新角色的描述。
13. 在 Step 1: Select trusted entities（步骤 1：选择可信实体）或 Step 2: Add permissions（步骤 2：添加权限）部分中的 Edit（编辑），以编辑角色的用户案例和权限。您将返回之前的页面进行编辑。
14. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
15. 检查角色，然后选择创建角色。

> **Important**
>
> 请注意，这只是所需配置的前半部分。您还必须向受信任账户中的各个用户授予在控制台中切换到此角色或以编程方式担任此角色的权限。有关这一步骤的更多信息，请参阅向用户授予切换角色的权限。

创建 IAM 角色 (Amazon CLI)

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。（可选）您还可以为您的角色设置权限边界。

创建用于跨账户存取的角色（Amazon CLI）

1. 创建角色：aws iam create-role
2. 将托管权限策略附加到角色：aws iam attach-role-policy

   或者

   为角色创建内联权限策略：aws iam put-role-policy
3. （可选）通过附加标签来向角色添加自定义属性：aws iam tag-role

   有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。
4. （可选）为角色设置权限边界：aws iam put-role-permissions-boundary

   权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

以下示例演示了在简单环境中创建跨账户角色的前两个最常见的步骤。此示例允许 123456789012 账户中的任何用户担任角色并查看 example_bucket Amazon S3 存储桶。该示例还假设您使用运行 Windows 的客户端计算机，并且已使用您的账户凭证和区域配置了命令行界面。有关更多信息，请参阅配置 Amazon 命令行界面。

在此示例中，在您创建角色时，在第一个命令中包括以下信任策略。此信任策略允许 123456789012 账户中的用户使用 AssumeRole 操作担任角色，但前提是用户使用 SerialNumber 和 TokenCode 参数提供 MFA 身份验证。有关 MFA 的更多信息，请参阅 IAM 中的 Amazon 多重身份验证。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
          "Action": "sts:AssumeRole",
          "Condition": { "Bool": { "aws:MultiFactorAuthPresent": "true" } }
      }
  ]
}
```

> **Important**
>
> 如果 Principal 元素中包含特定 IAM 角色或用户的 ARN，在保存策略时该 ARN 将转换为唯一的主体 ID。如果有人希望通过删除并重新创建角色或用户来提升权限，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为 ARN。但是，如果您删除角色或用户，则主体 ID 会显示在控制台中，因为 Amazon 无法再将其映射回 ARN。因此，如果您删除并重新创建了信任策略的 Principal 元素所引用的用户或角色，您必须编辑角色以替换 ARN。

当您使用第二个命令时，您必须将现有托管策略附加到角色。下面的权限策略仅允许担任角色的任何人对 example_bucket Amazon S3 存储桶执行 ListBucket 操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
      {
          "Effect": "Allow",
          "Action": "s3:ListBucket",
          "Resource": "arn:aws:s3:::example_bucket"
      }
  ]
}
```

要创建此 Test-UserAccess-Role 角色，您必须先将名为 trustpolicyforacct123456789012.json 的上述信任策略保存到本地 policies 驱动器中的 C: 文件夹。然后，使用名称 PolicyForRole 将上述权限策略保存为您的 Amazon Web Services 账户 中的客户管理型策略。然后您可以使用以下命令来创建角色并附加托管策略。

```
# Create the role and attach the trust policy file that allows users in the specified
 account to assume the role.
$ aws iam create-role --role-name Test-UserAccess-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforacct123456789012.json

# Attach the permissions policy (in this example a managed policy) to the role to
 specify what it is allowed to do.
```

```
$ aws iam attach-role-policy --role-name Test-UserAccess-Role --policy-arn
 arn:aws:iam::123456789012:policy/PolicyForRole
```

> **Important**
>
> 请注意，这只是所需配置的前半部分。您还必须对受信任账户中的各个用户授予切换角色的权限。有关这一步骤的更多信息，请参阅向用户授予切换角色的权限。

在创建角色并向该角色授予执行 Amazon 任务或访问 Amazon 资源的权限后，123456789012 账户中的任何用户都可以担任该角色。有关更多信息，请参阅 切换到 IAM 角色（Amazon CLI）。

创建 IAM 角色 (Amazon API)

从 Amazon API 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 API 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。（可选）您还可以为您的角色设置权限边界。

在代码中创建角色 (Amazon API)

1. 创建角色：CreateRole

   您可以指定一个文件位置作为角色的信任策略。
2. 将托管权限策略附加到角色：AttachRolePolicy

   或者

   为角色创建内联权限策略：PutRolePolicy

   > **Important**
   >
   > 请注意，这只是所需配置的前半部分。您还必须对受信任账户中的各个用户授予切换角色的权限。有关这一步骤的更多信息，请参阅向用户授予切换角色的权限。
3. （可选）通过附加标签来向角色添加自定义属性：TagRole

   有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。
4. （可选）为角色设置权限边界：PutRolePermissionsBoundary

   权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

在创建角色并向该角色授予执行 Amazon 任务或访问 Amazon 资源的权限后，您必须向账户中的用户授予允许他们担任该角色的权限。有关担任角色的更多信息，请参阅 切换到 IAM 角色（Amazon API）。

创建 IAM 角色 (Amazon CloudFormation)

有关在 Amazon CloudFormation 中创建 IAM 角色的信息，请参阅 Amazon CloudFormation 用户指南中的资源和属性参考和示例。

有关 Amazon CloudFormation 中的 IAM 模板的更多信息，请参阅Amazon CloudFormation用户指南中的 Amazon Identity and Access Management 模板代码段。

创建向 Amazon 服务委派权限的角色

许多 Amazon 服务要求您使用角色来允许该服务代表您访问其他服务中的资源。由一项服务担任、代表您执行操作的角色称为服务角色。当某角色为某项服务提供专门用途时，它会被归类为服务相关角色。要查看哪些服务支持使用服务相关角色，或服务是否支持任何形式的临时凭证，请参阅 使用 IAM 的 Amazon 服务。要了解单个服务如何使用角色，请选择表格中的服务名称以查看该服务对应的文档。

设置 `PassRole` 权限时，应确保用户所传递角色的权限不会超过您希望该用户拥有的权限。例如，可能不允许 Alice 执行任何 Amazon S3 操作。如果 Alice 可以将角色传递给允许 Amazon S3 操作的服务，则该服务可以在执行作业时代表 Alice 执行 Amazon S3 操作。

有关如何通过角色委派权限的信息，请参阅角色术语和概念。

服务角色权限

您必须配置权限以允许 IAM 实体（用户或角色）创建或编辑服务角色。

> **Note**
>
> 服务相关角色的 ARN 包括服务主体，它在以下策略中显示为 *`SERVICE-NAME`*`.amazonaws.com`。请勿尝试猜测服务主体，因为它区分大小写，并且格式会因 Amazon 服务而异。要查看服务的服务主体，请参阅其服务相关角色文档。

允许 IAM 实体创建特定服务角色

将以下策略添加到需要创建服务角色的 IAM 实体中。此策略允许您为指定服务创建具有特定名称的服务角色。然后，可将托管或内联策略附加到该角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:CreateRole",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        }
    ]
}
```

允许 IAM 实体创建任何服务角色

Amazon 建议您只允许管理用户创建任何服务角色。拥有创建角色和附加任何策略权限的人员可以升级自己的权限。相反，应创建一个策略，允许他们仅创建他们所需的角色，或者让管理员代表他们创建服务角色。

要附加允许管理员访问您整个 Amazon Web Services 账户 的策略，请使用 AdministratorAccess Amazon 托管策略。

允许 IAM 实体编辑服务角色

将以下策略添加到需要编辑服务角色的 IAM 实体中。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EditSpecificServiceRole",
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
```

```
                "iam:DeleteRolePolicy",
                "iam:DetachRolePolicy",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:ListAttachedRolePolicies",
                "iam:ListRolePolicies",
                "iam:PutRolePolicy",
                "iam:UpdateRole",
                "iam:UpdateRoleDescription"
            ],
            "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
        },
        {
            "Sid": "ViewRolesAndPolicies",
            "Effect": "Allow",
            "Action": [
                "iam:GetPolicy",
                "iam:ListRoles"
            ],
            "Resource": "*"
        }
    ]
}
```

允许 IAM 实体删除特定服务角色

将以下语句添加到需要删除指定服务角色的 IAM 实体的权限策略。

```
{
    "Effect": "Allow",
    "Action": "iam:DeleteRole",
    "Resource": "arn:aws:iam::*:role/SERVICE-ROLE-NAME"
}
```

允许 IAM 实体删除任何服务相关角色

Amazon 建议您只允许管理用户删除任何服务角色。相反，应创建一个策略，允许他们仅删除他们所需的角色，或者让管理员代表他们删除服务角色。

要附加到允许管理员访问您整个 Amazon Web Services 账户 的策略，请使用 AdministratorAccess Amazon 托管策略。

创建用于 Amazon 服务的角色（控制台）

您可使用 Amazon Web Services 管理控制台创建用于服务的角色。由于某些服务支持多个服务角色，请参阅您的服务的 Amazon 文档以查看要选择的使用案例。您可以了解如何为角色分配必要的信任策略和权限策略，以便服务能够代表您担任角色。可用于控制您的角色的权限的步骤可能会有所不同，具体取决于服务如何定义使用案例，以及您是否创建了服务相关角色。

Console

创建用于 Amazon Web Services 服务 的角色（IAM 控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 Amazon Web Services 服务。
4. 对于服务或使用案例，请选择服务，然后选择使用案例。用例由服务定义以包含服务要求的信任策略。
5. 选择下一步。
6. 对于权限策略，选项取决于您选择的使用案例：
   - 如果服务定义了角色的权限，则您无法选择权限策略。
   - 从一组有限的权限策略中进行选择。
   - 从所有权限策略中进行选择。
   - 不选择任何权限策略，创建角色后创建策略，然后将这些策略附加到该角色。
7. （可选）设置权限边界。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。
   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包括您的账户中的 Amazon 托管式策略和客户管理型策略的列表。
   b. 选择要用于权限边界的策略。
8. 选择下一步。
9. 对于角色名称，选项取决于服务：
   - 如果服务定义角色名称，则您无法编辑角色名称。
   - 如果服务定义角色名称的前缀，您可以输入可选的后缀。
   - 如果服务未定义角色名称，您可以为该角色命名。

> **Important**
>
> 命名角色时，请注意以下事项：
>
> - 角色名称在您的 Amazon Web Services 账户 中必须是唯一的，且不能因大小写而变得唯一。
>
> 例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。
> - 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。

10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。
12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。有关在 IAM 中使用标签的更多信息，请参阅《IAM 用户指南》中的 Amazon Identity and Access Management 资源的标签。
13. 检查该角色，然后选择创建角色。

创建用于服务的角色 (Amazon CLI)

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。如果您使用的服务是 Amazon EC2，则还必须创建实例配置文件并向其添加角色。（可选）您还可以为您的角色设置权限边界。

从 Amazon CLI 为 Amazon 服务创建角色

1. 以下 `create-role` 命令创建了一个名为 Test-Role 的角色并向其附加了信任策略：

   ```
   aws iam create-role --role-name Test-Role --assume-role-policy-document
   file://Test-Role-Trust-Policy.json
   ```

2. 将托管权限策略附加到角色：aws iam attach-role-policy。

   例如，以下内容 `attach-role-policy` 命令将名为 `ReadOnlyAccess` 的附加 Amazon 托管策略附加到了名为 `ReadOnlyRole` 的 IAM 角色：

```
aws iam attach-role-policy --policy-arn arn:aws:iam::aws:policy/
ReadOnlyAccess --role-name ReadOnlyRole
```

或者

为角色创建内联权限策略：aws iam put-role-policy

要添加内联权限策略，请参阅以下示例：

```
aws iam put-role-policy --role-name Test-Role --policy-name
ExamplePolicy --policy-document file://AdminPolicy.json
```

3. （可选）通过附加标签来向角色添加自定义属性：aws iam tag-role

   有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。

4. （可选）为角色设置权限边界：aws iam put-role-permissions-boundary

   权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

如果要将角色与 Amazon EC2 或使用 Amazon EC2 的其他 Amazon 服务结合使用，则必须将角色存储在实例配置文件中。实例配置文件是一个角色容器，可在启动时附加到 Amazon EC2 实例。一个实例配置文件只能包含一个 角色，不能提高该限制。如果使用 Amazon Web Services 管理控制台创建角色，则会为您创建具有与角色相同的名称的实例配置文件。有关实例配置文件的更多信息，请参阅使用实例配置文件。有关如何通过角色启动 EC2 实例的信息，请参阅《Amazon EC2 用户指南》中的控制对 Amazon EC2 资源的访问。

创建实例配置文件并在其中存储角色 (Amazon CLI)

1. 创建实例配置文件：aws iam create-instance-profile
2. 向实例配置文件添加角色：aws iam add-role-to-instance-profile

以下 Amazon CLI 示例命令集演示了创建角色并附加权限的前两个步骤。它还演示了创建实例配置文件并将角色添加到该配置文件中的两个步骤。此示例信任策略允许 Amazon EC2 服务担任角色并查看 `example_bucket` Amazon S3 存储桶。该示例还假设您使用运行 Windows 的客户端计算机，并且已使用您的账户凭证和区域配置了命令行界面。有关更多信息，请参阅配置 Amazon 命令行界面。

在此示例中，在您创建角色时，在第一个命令中包括以下信任策略。此信任策略允许 Amazon EC2 服务代入角色。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"Service": "ec2.amazonaws.com"},
    "Action": "sts:AssumeRole"
  }
}
```

当您使用第二个命令时，您必须将权限策略附加到角色。下面的示例权限策略仅允许角色对 example_bucket Amazon S3 存储桶执行 ListBucket 操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
  }
}
```

要创建此 Test-Role-for-EC2 角色，您必须先将上述名为 trustpolicyforec2.json 的信任策略和上述名为 permissionspolicyforec2.json 的权限策略保存到您的本地 C: 驱动器中的 policies 目录。然后，您可以使用以下命令来创建角色，附加策略，创建实例配置文件，并将角色添加到实例配置文件中。

```
# Create the role and attach the trust policy that allows EC2 to assume this role.
$ aws iam create-role --role-name Test-Role-for-EC2 --assume-role-policy-document
 file://C:\policies\trustpolicyforec2.json

# Embed the permissions policy (in this example an inline policy) to the role to
 specify what it is allowed to do.
$ aws iam put-role-policy --role-name Test-Role-for-EC2 --policy-name Permissions-
Policy-For-Ec2 --policy-document file://C:\policies\permissionspolicyforec2.json
```

```
# Create the instance profile required by EC2 to contain the role
$ aws iam create-instance-profile --instance-profile-name EC2-ListBucket-S3

# Finally, add the role to the instance profile
$ aws iam add-role-to-instance-profile --instance-profile-name EC2-ListBucket-S3 --
role-name Test-Role-for-EC2
```

启动 EC2 实例时，如果您使用 Amazon 控制台，请在 Configure Instance Details (配置实例详细信息) 页面中指定实例配置文件名称。如果使用 `aws ec2 run-instances` CLI 命令，请指定 `--iam-instance-profile` 参数。

创建用于服务的角色 (Amazon API)

从 Amazon API 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 API 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。如果您使用的服务是 Amazon EC2，则还必须创建实例配置文件并向其添加角色。（可选）您还可以为您的角色设置权限边界。

为 Amazon 服务创建角色 (Amazon API)

1. 创建角色：CreateRole

   您可以指定一个文件位置作为角色的信任策略。

2. 将托管权限策略附加到角色：AttachRolePolicy

   或者

   为角色创建内联权限策略：PutRolePolicy

3. （可选）通过附加标签来向角色添加自定义属性：TagRole

   有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。

4. （可选）为角色设置权限边界：PutRolePermissionsBoundary

   权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

如果要将角色与 Amazon EC2 或使用 Amazon EC2 的其他 Amazon 服务结合使用，则必须将角色存储在实例配置文件中。实例配置文件是角色的容器。每个实例配置文件只能包含一个角色，不能提高该限制。如果在 Amazon Web Services 管理控制台中创建角色，则为您创建具有与角色相同的名称

的实例配置文件。有关实例配置文件的更多信息，请参阅使用实例配置文件。有关如何通过角色启动 Amazon EC2 实例的信息，请参阅《Amazon EC2 用户指南》中的控制对 Amazon EC2 资源的访问。

创建实例配置文件并在其中存储角色 (Amazon API)

1. 创建实例配置文件：CreateInstanceProfile
2. 向实例配置文件添加角色：AddRoleToInstanceProfile

创建服务相关角色

服务相关角色是一种独特类型的 IAM 角色，它与Amazon服务直接相关。服务相关角色由服务预定义，具有服务代表您调用其他 Amazon 服务所需的所有权限。相关的服务还定义了创建、修改和删除服务相关角色的方式。服务可以自动创建或删除角色。它可能允许您在服务的向导或流程中创建、修改或删除角色。或者，它可能需要您使用 IAM 创建或删除角色。无论使用哪种方法，服务相关角色都可让您简化 服务的设置流程，因为您不必手动添加服务代表您完成操作所需的权限。

> **Note**
>
> 请记住，服务角色不同于服务相关角色。服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务 委派权限的角色。服务关联角色是一种与 Amazon Web Services 服务 关联的服务角色。服务可以代入代表您执行操作的角色。服务关联角色显示在您的 Amazon Web Services 账户 中，并由该服务拥有。IAM 管理员可以查看但不能编辑服务关联角色的权限。

链接服务会定义其服务相关角色的权限，除非另外定义，否则仅该服务可以担任角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其他 IAM 实体。

您必须先删除角色的相关资源，之后才能删除角色。这有助于防止您无意中删除访问这些资源的权限。

> **Tip**
>
> 有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

服务相关角色权限

您必须为 IAM 实体（用户、组或角色）配置权限，以允许用户或角色创建或编辑服务相关角色。

> **Note**
>
> 服务相关角色的 ARN 包括服务主体，它在以下策略中显示为 *SERVICE-NAME*.amazonaws.com。请勿尝试猜测服务主体，因为它区分大小写，并且格式会因 Amazon 服务而异。要查看服务的服务主体，请参阅其服务相关角色文档。

允许 IAM 实体创建特定服务相关角色

将以下策略添加到需要创建服务相关角色的 IAM 实体中。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*",
            "Condition": {"StringLike": {"iam:AWSServiceName": "SERVICE-NAME.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:AttachRolePolicy",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
        }
    ]
}
```

允许 IAM 实体创建任何服务相关角色

将以下语句添加到 IAM 实体的权限策略，该实体需要创建服务相关角色或任何包含所需策略的服务角色。此策略语句不允许 IAM 实体将策略附加到该角色。

```
{
    "Effect": "Allow",
    "Action": "iam:CreateServiceLinkedRole",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体编辑任何服务角色的描述

将以下语句添加到 IAM 实体的权限策略，该实体需要编辑服务相关角色或任何服务角色的描述。

```
{
    "Effect": "Allow",
    "Action": "iam:UpdateRoleDescription",
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体删除特定服务相关角色

将以下语句添加到需要删除服务相关角色的 IAM 实体的权限策略。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/SERVICE-
NAME.amazonaws.com/SERVICE-LINKED-ROLE-NAME-PREFIX*"
}
```

允许 IAM 实体删除任何服务相关角色

将以下语句添加到 IAM 实体的权限策略，该实体需要删除服务相关角色，而不需要删除服务角色。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:DeleteServiceLinkedRole",
```

```
        "iam:GetServiceLinkedRoleDeletionStatus"
    ],
    "Resource": "arn:aws:iam::*:role/aws-service-role/*"
}
```

允许 IAM 实体将现有角色传递给服务

有些 Amazon 服务允许您将现有角色传递给服务，而不是创建新的服务相关角色。为此，用户必须具有将角色传递给服务的权限。将以下语句添加到需要传递角色的 IAM 实体的权限策略。此策略语句还允许实体查看可从中选择要传递的角色的角色列表。有关更多信息，请参阅 向用户授予权限以将角色传递给 Amazon 服务。

```
{
  "Sid": "PolicyStatementToAllowUserToListRoles",
  "Effect": "Allow",
  "Action": ["iam:ListRoles"],
  "Resource": "*"
},
{
  "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
  "Effect": "Allow",
  "Action": [ "iam:PassRole" ],
  "Resource": "arn:aws:iam::account-id:role/my-role-for-XYZ"
}
```

使用服务相关角色的间接权限

服务相关角色授予的权限可以间接转让给其他用户和角色。Amazon 服务使用某个服务相关角色时，该服务相关角色可以使用自己的权限调用其他 Amazon 服务。这意味着，如果用户和角色有权调用使用某个服务相关角色的服务，将能够间接访问该服务相关角色可以访问的服务。

例如，在创建 Amazon RDS 数据库实例时，RDS 的服务相关角色如果尚不存在，则会自动创建。借助该服务相关角色，RDS 将可以代表您调用 Amazon EC2、Amazon SNS、Amazon CloudWatch Logs 和 Amazon Kinesis 等服务。如果您允许账户中的用户和角色修改或创建 RDS 数据库，则这些用户和角色将可以通过调用 RDS，从而与 Amazon EC2、Amazon SNS、Amazon CloudWatch Logs 日志和 Amazon Kinesis 资源进行间接交互，因为 RDS 会使用其服务相关角色来访问这些资源。

创建服务相关角色的方法

您用来创建服务相关角色的方法取决于服务。在某些情况下，无需手动创建服务相关角色。例如，在服务中完成特定操作 (如创建资源) 时，服务可能为您创建服务相关角色。或者，如果您在某项服务开始

支持服务相关角色之前已在使用该服务，则该服务可能自动在您的账户中创建角色。要了解更多信息，请参阅我的 Amazon 账户中出现新角色。

在其他情况下，服务可能支持使用服务控制台、API 或 CLI 手动创建服务相关角色。有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。要了解服务是否支持创建服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。

如果服务不支持创建角色，则可以使用 IAM 创建服务相关角色。

> **Important**
>
> 服务相关角色将计入您的 Amazon Web Services 账户 中的 IAM 角色限制，但如果您已达到限制，仍可以在账户中创建服务相关角色。只有服务相关角色可以超过此限制。

创建服务相关角色（控制台）

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色。此外，需了解您是否能从该服务的控制台、API 或 CLI 创建角色。

创建服务相关角色（控制台）

1. 登录 Amazon Web Services 管理控制台，打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。然后，选择创建角色。
3. 选择Amazon 服务角色类型。
4. 选择用于您的服务的用例。使用案例由服务定义以包含服务所需的信任策略。然后选择下一步。
5. 选择一个或多个要附加到角色的权限策略。根据您选择的使用案例，服务可能执行以下任意操作：
   - 定义角色所使用的权限。
   - 允许您从一组有限的权限中进行选择。
   - 允许您从任意权限中进行选择。
   - 允许您此时不选择策略，稍后创建策略，然后将这些策略附加到角色。

   选中分配您希望角色具有的权限的策略旁边的复选框，然后选择下一步。

> **Note**
>
> 使用角色的任何实体都可以使用您指定的权限。默认情况下，角色没有权限。

6. 对于角色名称，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并让您输入可选的后缀。

   如果可能，请输入要添加到默认名称的角色名称后缀。该后缀可帮助您确定角色的用途。角色名称在您的 Amazon 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **`<service-linked-role-name>_SAMPLE`** 和 **`<service-linked-role-name>_sample`** 的角色。由于多个单位可能引用该角色，角色创建完毕后无法编辑角色名称。
7. （可选）对于 Description（描述），编辑新服务相关角色的描述。
8. 您无法在创建过程中将标签附加到服务相关角色。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
9. 检查角色，然后选择创建角色。

创建服务相关角色(Amazon CLI)

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色，以及您是否能从该服务的 CLI 创建角色。如果服务 CLI 不受支持，您可以使用 IAM 命令创建具有服务担任角色时所需的信任策略和内联策略的服务相关角色。

创建服务相关角色 (Amazon CLI)

运行如下命令：

```
aws iam create-service-linked-role --aws-service-name SERVICE-NAME.amazonaws.com
```

创建服务相关角色 (Amazon API)

在 IAM 中创建服务相关角色之前，请查明链接服务是否已自动创建服务相关角色，以及您是否能从该服务的 API 创建角色。如果服务 API 不受支持，您可以使用 Amazon API 创建具有服务代入角色时所需的信任策略和内联策略的服务相关角色。

创建服务相关角色 (Amazon API)

使用 CreateServiceLinkedRole API 调用。在请求中，指定 *SERVICE_NAME_URL*.amazonaws.com 的服务名称。

例如，要创建 Lex Bots 服务相关角色，请使用 `lex.amazonaws.com`。

为第三方身份提供者创建角色

您可以使用身份提供程序而不必在 Amazon Web Services 账户 中创建 IAM 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合和身份提供程序的更多信息，请参阅身份提供程序和 Amazon 中的联合身份验证。

为 OIDC 和 SAML 联合主体创建角色（控制台）

创建角色的过程取决于您选择的第三方提供商：

- 有关 OpenID Connect（OIDC），请参阅 创建用于 OpenID Connect 联合身份验证（控制台）的角色。
- 有关 SAML 2.0，请参阅创建用于 SAML 2.0 联合身份验证的角色（控制台）。

为联合访问创建角色 (Amazon CLI)

从 Amazon CLI 中为支持的身份提供程序（OIDC 或 SAML）创建角色的步骤是相同的。区别在于，您在先决条件步骤中创建的信任策略的内容不同。首先，按照先决条件部分中针对您正在使用的提供商类型的步骤操作：

- 有关 OIDC 提供商，请参阅创建用于 OIDC 的角色的先决条件。
- 有关 SAML 提供商，请参阅创建用于 SAML 的角色的先决条件。

从 Amazon CLI 创建角色涉及几个步骤。当您使用控制台创建角色时，很多步骤会自动完成，但是使用 Amazon CLI 时，您必须自行完成每一个步骤。您必须创建角色，然后为角色分配权限策略。（可选）您还可以为您的角色设置权限边界。

创建角色 (Amazon CLI)

1. 创建角色：aws iam create-role
2. 将权限策略附加到角色：aws iam attach-role-policy

   或者

   为角色创建内联权限策略：aws iam put-role-policy
3. （可选）通过附加标签来向角色添加自定义属性：aws iam tag-role

有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。

4. （可选）为角色设置权限边界：aws iam put-role-permissions-boundary

   权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

以下示例演示了在简单环境中创建身份提供程序角色的前两个最常见的步骤。此示例允许 123456789012 账户中的任何用户担任角色并查看 example_bucket Amazon S3 存储桶。该示例还假设您在运行 Windows 的计算机上运行 Amazon CLI，并且已使用您的凭证配置了 Amazon CLI。有关更多信息，请参阅配置 Amazon Command Line Interface。

以下示例信任策略是为用户使用 Amazon Cognito 登录时的移动应用程序设计的。在此示例中，*us-east:12345678-ffff-ffff-ffff-123456* 表示由 Amazon Cognito 分配的身份池 ID。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "RoleForCognito",
        "Effect": "Allow",
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud": "us-
east:12345678-ffff-ffff-ffff-123456"}}
    }
}
```

下面的权限策略仅允许担任角色的任何人对 example_bucket Amazon S3 存储桶执行 ListBucket 操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::example_bucket"
```

```
  }
}
```

要创建此 Test-Cognito-Role 角色，您必须首先将上述名为 trustpolicyforcognitofederation.json 的信任策略和上述名为 permspolicyforcognitofederation.json 的权限策略保存到您的本地 policies 驱动器中的 C: 文件夹。然后您可以使用以下命令来创建角色并附加内联策略。

```
# Create the role and attach the trust policy that enables users in an account to
 assume the role.
$ aws iam create-role --role-name Test-Cognito-Role --assume-role-policy-document
 file://C:\policies\trustpolicyforcognitofederation.json

# Attach the permissions policy to the role to specify what it is allowed to do.
aws iam put-role-policy --role-name Test-Cognito-Role --policy-name
 Perms-Policy-For-CognitoFederation --policy-document file://C:\policies
\permspolicyforcognitofederation.json
```

为联合访问创建角色 (Amazon API)

从 Amazon CLI 中为支持的身份提供程序（OIDC 或 SAML）创建角色的步骤是相同的。区别在于，您在先决条件步骤中创建的信任策略的内容不同。首先，按照先决条件部分中针对您正在使用的提供商类型的步骤操作：

- 有关 OIDC 提供商，请参阅创建用于 OIDC 的角色的先决条件。
- 有关 SAML 提供商，请参阅创建用于 SAML 的角色的先决条件。

要创建角色（Amazon API）

1. 创建角色：CreateRole
2. 将权限策略附加到角色：AttachRolePolicy

   或者

   为角色创建内联权限策略：PutRolePolicy
3. （可选）通过附加标签来向角色添加自定义属性：TagRole

   有关更多信息，请参阅 管理 IAM 角色（Amazon CLI 或 Amazon API）的标签。
4. （可选）为角色设置权限边界：PutRolePermissionsBoundary

权限边界控制角色可以具有的最大权限。权限边界是一项高级 Amazon 功能。

创建用于 OpenID Connect 联合身份验证（控制台）的角色

您可以使用 OpenID Connect (OIDC) 联合身份验证身份提供者，而不必在 Amazon Web Services 账户 中创建 Amazon Identity and Access Management 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合身份验证和 IdP 的更多信息，请参阅 身份提供程序和 Amazon 中的联合身份验证。

创建用于 OIDC 的角色的先决条件

您必须先完成以下先决条件步骤，然后才能创建用于 OIDC 联合身份验证的角色。

准备创建用于 OIDC 联合身份验证的角色

1. 注册一项或多项提供 OIDC 联合身份的服务。如果创建需要访问 Amazon 资源的应用程序，则还需要使用提供商信息来配置应用程序。当您执行此操作时，提供程序会为您提供一个应用程序 ID 或受众 ID，该 ID 在您的应用程序中具有唯一性。（不同的提供程序使用不同的术语表示此过程。此指南使用配置一词表示通过提供程序标识应用程序的过程。）可以对每个提供商配置多个应用程序，也可以对单个应用程序配置多个提供商。查看有关使用身份提供程序的信息，如下所示：
   - Login with Amazon 开发人员中心
   - 在 Facebook 开发人员网站上，将 Facebook 登录名添加到您的应用程序或网站。
   - 在 Google 开发人员网站上，使用 OAuth 2.0 进行登录 (OpenID Connect)。
2. 从 IdP 接收必要信息后，在 IAM 中创建 IdP。有关更多信息，请参阅 在 IAM 中创建 OpenID Connect（OIDC）身份提供者。

   > **Important**
   >
   > 如果您使用的是 Google、Facebook 或 Amazon Cognito 的 OIDC IdP，请勿在 Amazon Web Services 管理控制台 中创建单独的 IAM IdP。这些 OIDC 身份提供程序已内置到 Amazon，并可供您使用。跳过此步骤并在以下步骤中使用 IdP 创建新角色。

3. 为已进行 IdP 身份验证的用户要担任的角色准备策略。正如任何角色一样，移动应用程序的角色包含两个策略。一个是指定谁可以担任角色的信任策略。另一个是指定允许或拒绝移动应用程序访问的 Amazon 操作和资源的权限策略。

对于 web IdP，我们建议您使用 Amazon Cognito 来管理身份。在这种情况下，请使用类似于以下示例的信任策略。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"Federated": "cognito-identity.amazonaws.com"},
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {
            "StringEquals": {"cognito-identity.amazonaws.com:aud": "us-east-2:12345678-abcd-abcd-abcd-123456"},
            "ForAnyValue:StringLike": {"cognito-identity.amazonaws.com:amr": "unauthenticated"}
        }
    }
}
```

将 us-east-2:12345678-abcd-abcd-abcd-123456 替换为由 Amazon Cognito 分配给您的身份池 ID。

如果您手动配置 OIDC IdP，则在创建信任策略时，必须使用三个值来确保只有您的应用程序可以担任此角色：

- 对于 Action 元素，使用 sts:AssumeRoleWithWebIdentity 操作。
- 对于 Principal 元素，使用字符串 {"Federated":*providerUrl/providerArn*}。
  - 对于一些常见的 OIDC IdP，*providerUrl* 是 URL。以下示例包括为一些常见 IdP 指定主体的方法：

    "Principal":{"Federated":"cognito-identity.amazonaws.com"}

    "Principal":{"Federated":"www.amazon.com"}

    "Principal":{"Federated":"graph.facebook.com"}

    "Principal":{"Federated":"accounts.google.com"}

- 对于其他的 OIDC 提供程序，请使用您在 Step 2 中创建的 OIDC IdP 的 Amazon 资源名称（ARN），如下例所示：

  ```
  "Principal":{"Federated":"arn:aws:iam::123456789012:oidc-provider/
  server.example.com"}
  ```

- 对于 Condition 元素，使用 StringEquals 条件来限制权限。测试身份池 ID（对于 Amazon Cognito）或应用程序 ID（对于其他提供商）。身份池 ID 应与您通过 IdP 配置应用程序时获得的应用程序 ID 一致。ID 一致可确保请求来自您的应用程序。

  > **Note**
  >
  > Amazon Cognito 身份池的 IAM 角色信任服务主体 cognito-identity.amazonaws.com 担任该角色。此类角色必须包含至少一个条件键来限制可以担任该角色的主体。
  > 其他注意事项适用于承担跨账户 IAM 角色的 Amazon Cognito 身份池。这些角色的信任策略必须接受 cognito-identity.amazonaws.com 服务主体，并且必须包含 aud 条件键，以限制预期身份池中的用户担任角色。如果不满足此条件，则信任 Amazon Cognito 身份池的策略会带来风险，即来自非预期身份池的用户可能担任该角色。有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的基本（经典）身份验证中的 IAM 角色的信任策略。

  根据您所使用的 IdP 创建类似于以下示例之一的条件元素：

  ```
  "Condition": {"StringEquals": {"cognito-identity.amazonaws.com:aud":
  "us-east:12345678-ffff-ffff-ffff-123456"}}
  ```

  ```
  "Condition": {"StringEquals": {"www.amazon.com:app_id":
  "amzn1.application-oa2-123456"}}
  ```

  ```
  "Condition": {"StringEquals": {"graph.facebook.com:app_id":
  "111222333444555"}}
  ```

  ```
  "Condition": {"StringEquals": {"accounts.google.com:aud":
  "66677788899900pro0"}}
  ```

  对于 OIDC 提供商，请将 OIDC IdP 的完全限定 URL 与 aud 上下文密钥一起使用，如以下示例所示：

```
"Condition": {"StringEquals": {"server.example.com:aud":
"appid_from_oidc_idp"}}
```

> **Note**
>
> 角色的信任策略中的主体的值将因 IdP 而异。用于 OIDC 的角色只能指定一个主体。因此，如果移动应用程序允许用户从多个 IdP 登录，请为您想要支持的每个 IdP 创建单独角色。为每个 IdP 创建单独的信任策略。

如果用户使用移动应用程序从 Login with Amazon 登录，则将应用以下示例信任策略。在示例中，*amzn1.application-oa2-123456* 表示使用 Login with Amazon 配置应用程序时由 Amazon 分配的应用程序 ID。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
          "Sid": "RoleForLoginWithAmazon",
          "Effect": "Allow",
          "Principal": {"Federated": "www.amazon.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"www.amazon.com:app_id":
 "amzn1.application-oa2-123456"}}
      }]
  }
```

如果用户使用移动应用程序从 Facebook 登录，则将应用以下示例信任策略。在此示例中，*111222333444555* 表示由 Facebook 分配的应用程序 ID。

JSON

```
{
      "Version":"2012-10-17",
```

```
      "Statement": [{
          "Sid": "RoleForFacebook",
          "Effect": "Allow",
          "Principal": {"Federated": "graph.facebook.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"graph.facebook.com:app_id":
 "111222333444555"}}
      }]
  }
```

如果用户使用移动应用程序从 Google 登录，则将应用以下示例信任策略。在此示例中，*666777888999000* 表示由 Google 分配的应用程序 ID。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
          "Sid": "RoleForGoogle",
          "Effect": "Allow",
          "Principal": {"Federated": "accounts.google.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"accounts.google.com:aud":
 "666777888999000"}}
      }]
  }
```

如果用户使用移动应用程序从 Amazon Cognito 登录，则将应用以下示例信任策略。在此示例中，*us-east:12345678-ffff-ffff-ffff-123456* 表示由 Amazon Cognito 分配的身份池 ID。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
          "Sid": "RoleForCognito",
```

```
          "Effect": "Allow",
          "Principal": {"Federated": "cognito-identity.amazonaws.com"},
          "Action": "sts:AssumeRoleWithWebIdentity",
          "Condition": {"StringEquals": {"cognito-
identity.amazonaws.com:aud": "us-east:12345678-ffff-ffff-ffff-123456"}}
       }]
  }
```

创建用于 OIDC 的角色

完成先决条件后，您可在 IAM 中创建角色。对于公认的共享 OpenID Connect（OIDC）身份提供者（IdP），IAM 要求对名为身份提供者控制的 JSON Web 令牌（JWT）中的特定声明进行明确评估。有关哪些 OIDC IdP 拥有身份提供者控制的更多信息，请参阅 适用于共享 OIDC 提供者的身份提供者控制。

以下过程介绍了如何在 Amazon Web Services 管理控制台 中创建用于 OIDC 联合身份验证的角色。要从 Amazon CLI 或 Amazon API 创建角色，请参阅为第三方身份提供者创建角色中的过程。

> **Important**
>
> 如果您使用的是 Amazon Cognito，则使用 Amazon Cognito 控制台来设置角色。否则，请使用 IAM 控制台来创建用于 OIDC 联合身份验证的角色。

创建用于 OIDC 联合身份验证的 IAM 角色

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选择创建角色。
3. 选择 Web 身份作为可信实体类型，然后选择下一步。
4. 对于 Identity provider（身份提供程序），请为您的角色选择 IdP：
   - 如果要为单个 web IdP 创建角色，请选择 Login with Amazon、Facebook 或 Google。

> **Note**
>
> 您必须为想要支持的每个 IdP 创建单独的角色。

- 如果要为 Amazon Cognito 创建高级方案角色，请选择 Amazon Cognito。

> **Note**
>
> 您仅在处理高级方案时需要手动创建与 Amazon Cognito 一起使用的角色。否则，Amazon Cognito 可以为您创建角色。有关 Amazon Cognito 的更多信息，请参阅《Amazon Cognito 开发人员指南》中的身份池（联合身份）外部身份提供商。

- 要为 GitHub 操作创建角色，您需要首先将 GitHub OIDC 提供者添加到 IAM。将 GitHub OIDC 提供商添加到 IAM 后，选择 token.actions.githubusercont.com。

> **Note**
>
> 有关如何将 Amazon 配置为联合身份以信任 GitHub 的 OIDC 提供商的信息，请参阅 GitHub 文档 – 在 Amazon Web Services 中配置 OpenID Connect。有关限制与 GitHub IAM IdP 相关的角色访问权限的最佳做法的信息，请参阅此页面上的为 GitHub OIDC 身份提供程序配置角色。

- 要为 HashiCorp Cloud Platform（HCP）Terraform 创建角色，您首先必须将 Terraform OIDC 提供商添加到 IAM。将 Terraform OIDC 提供商添加到 IAM 后，选择 app.terraform.io。

> **Important**
>
> HashiCorp Cloud Platform（HCP）Terraform OIDC 提供商的 IAM 角色必须评估角色信任策略中的 IAM 条件键 `app.terraform.io:sub`。此条件键对哪些 HCP Terraform 组织、项目、工作区或运行阶段能够担任该角色作出限制。如果没有此条件键，则您的信任策略将根据组织外部的身份授予对您的角色和 Amazon 资源的访问权限，这不符合最低权限原则。
>
> 如果您为 Amazon 账户中与 HCP Terraform OIDC 提供商关联的角色设置或修改角色信任策略，但未评估 IAM 条件键 `app.terraform.io:sub`，则会收到错误。此外，如果您的角色信任策略未评估此条件键，则 Amazon STS 会拒绝授权请求。

5. 所请求的信息因您选择的 OIDC 提供者而异。

- 输入您的应用程序标识符。标识符的标签会根据所选提供程序发生变化：
  - 如果要为 Login with Amazon 创建角色，请在 Application ID（应用程序 ID）框中输入应用程序 ID。
  - 如果要为 Facebook 创建角色，请在 Application ID（应用程序 ID）框中输入应用程序 ID。

- 如果要为 Google 创建角色，请在 Audience（受众）框中输入受众名称。
- 如果要为 Amazon Cognito 创建角色，请在 Identity Pool ID（身份池 ID）框中输入您为 Amazon Cognito 应用程序创建的身份池的 ID。

- 如果您想为 GitHub Actions 创建角色，请输入以下详细信息：
  - 对于 Audience (受众)，请选择 `sts.amazonaws.com`。
  - 对于 GitHub 组织，输入 GitHub 组织名称。GitHub 组织名称为必填项，必须为包含短划线（-）的字母数字。您不能在 GitHub 组织名称中使用通配符（* 和？）。
  - （可选）对于 GitHub 存储库，输入 GitHub 存储库名称。如果不指定值，则默认为通配符（*）。
  - （可选）对于 GitHub 分支，输入 GitHub 分支名称。如果不指定值，则默认为通配符（*）。
- 如果您想为 HashiCorp Cloud Platform（HCP）Terraform 创建角色，则请输入以下详细信息：
  - 对于 Audience (受众)，请选择 `aws.workload.identity`。
  - 对于组织，输入组织名称。您可以为所有组织指定通配符（*）。
  - 对于项目，输入项目名称。您可以为所有项目指定通配符（*）。
  - 对于工作区，输入工作区名称。您可以为所有工作区指定通配符（*）。
  - 对于运行阶段，输入运行阶段名称。您可以为所有运行阶段指定通配符（*）。

6. （可选）对于条件（可选），选择添加条件以创建在应用程序的用户可以使用角色授予的权限之前必须满足的其他条件。例如，您可以添加一个条件，仅授权特定 IAM 用户 ID 访问 Amazon 资源。创建角色后，您还可以在信任策略中添加条件。有关更多信息，请参阅 更新角色信任策略。
7. 查看您的 OIDC 信息，然后选择下一步。
8. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或者选择 Create policy（创建策略）以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅 创建 IAM 策略。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选中您希望 OIDC 用户具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
9. （可选）设置权限边界。这是一项高级功能。

   打开 Permissions boundary（权限边界）部分，然后选择 Use a permissions boundary to control the maximum role permissions（使用权限边界控制最大角色权限）。选择要用于权限边界的策略。
10. 选择下一步。

11. 对于 Role name（角色名称），输入一个角色名称。角色名称在您的 Amazon Web Services 账户内必须是唯一的。不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能会引用此角色，因此您无法在创建角色后编辑角色名称。
12. （可选）对于 Description（描述），输入新角色的描述。
13. 要编辑角色的使用案例和权限，在 Step 1: Select trusted entities（步骤 1：选择可信实体）或 Step 2: Add permissions（步骤 2：添加权限）部分中选择 Edit（编辑）。
14. （可选）要向角色添加元数据，请以键值对的形式附加标签。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
15. 检查角色，然后选择创建角色。

为 GitHub OIDC 身份提供程序配置角色

如果将 GitHub 用作 OpenID Connect（OIDC）身份提供者（IdP），则最佳实践是限制可以代入与该 IAM IdP 关联的角色的实体。如果信任策略中包含条件语句，则可将角色限制为特定的 GitHub 组织、存储库或分支。可以使用条件键 `token.actions.githubusercontent.com:sub` 和字符串条件运算符来限制访问权限。我们建议将条件限制为您 GitHub 组织中的一组特定存储库或分支。有关如何将 Amazon 配置为联合身份以信任 GitHub 的 OIDC 的信息，请参阅 GitHub 文档 – 在 Amazon Web Services 中配置 OpenID Connect。

如果您在操作工作流或 OIDC 策略中使用 GitHub 环境，我们强烈建议您在环境中添加保护规则以提高安全性。使用部署分支和标签来限制可以部署到环境中的具体分支和标签。有关使用保护规则配置环境的更多信息，请参阅 GitHub 文章“使用环境进行部署”中的 部署分支和标签。

如果 GitHub 的 OIDC IdP 是角色的可信主体，IAM 会检查角色信任策略条件，以验证条件键 `token.actions.githubusercontent.com:sub` 是否存在并且其值并非单纯的通配符（* 和 ?）或者为空。创建或更新信任策略时，IAM 会执行此检查。如果条件键 `token.actions.githubusercontent.com:sub` 不存在，或者键值不符合上述值标准，则请求将会失败并返回错误。

> **Important**
>
> 如果未将条件键 `token.actions.githubusercontent.com:sub` 限制为特定的组织或存储库，则来自您控制范围之外的组织或存储库的 GitHub 操作可代入与您 Amazon 账户中的 GitHub IAM IdP 关联的角色。

以下示例信任策略限制对定义的 GitHub 组织、存储库和分支的访问。以下示例中的条件键 `token.actions.githubusercontent.com:sub` 的值是 GitHub 记录的默认主题值格式。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::012345678910:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub":
 "repo:GitHubOrg/GitHubRepo:ref:refs/heads/GitHubBranch"
        }
      }
    }
  ]
}
```

以下示例条件限制对定义的 GitHub 组织和存储库的访问，但授予对存储库内任何分支的访问权限。

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/GitHubRepo:*"
  }
}
```

以下示例条件限制对定义的 GitHub 组织内任何存储库或分支的访问。我们建议将条件键 `token.actions.githubusercontent.com:sub` 限制为一个特定的值，从而确保只能从 GitHub 组织内访问 GitHub 操作。

```
"Condition": {
  "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com"
  },
  "StringLike": {
    "token.actions.githubusercontent.com:sub": "repo:GitHubOrg/*"
  }
}
```

有关可用于策略中的条件检查的 OIDC 联合身份验证密钥的更多信息，请参阅 Amazon OIDC 联合身份验证的可用键。

创建用于 SAML 2.0 联合身份验证的角色（控制台）

您可以使用 SAML 2.0 联合身份验证而不必在 Amazon Web Services 账户 中创建 IAM 用户。利用身份提供者 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予访问您账户中的 Amazon 资源的权限。有关联合身份验证和身份提供者的更多信息，请参阅身份提供程序和 Amazon 中的联合身份验证。

> **Note**
>
> 为了提高联合身份验证弹性，我们建议您将 IdP 和Amazon联合身份验证配置为支持多个 SAML 登录端点。有关详细信息，请参阅 Amazon 安全博客文章如何使用区域性 SAML 端点进行失效转移。

创建用于 SAML 的角色的先决条件

您必须先完成以下先决条件步骤，然后才能创建用于 SAML 2.0 联合身份验证的角色。

准备创建用于 SAML 2.0 联合身份验证的角色

1. 在创建用于 SAML 联合身份验证的角色之前，必须在 IAM 中创建 SAML 提供商。有关更多信息，请参阅 在 IAM 中创建 SAML 身份提供者。
2. 为已进行 SAML 2.0 身份验证的用户要担任的角色准备策略。正如任何角色一样，用于 SAML 联合身份验证的角色包含两个策略。一个是指定谁可以代入角色的角色信任策略。另一个是指定允许或拒绝 SAML 联合主体访问的 Amazon 操作和资源的 IAM 权限策略。

   在为角色创建信任策略时，必须使用三个值来确保只有您的应用程序可以代入此角色：

- 对于 Action 元素，使用 sts:AssumeRoleWithSAML 操作。
- 对于 Principal 元素，使用字符串 {"Federated":*ARNofIdentityProvider*}。将 *ARNofIdentityProvider* 替换为您在Step 1 中创建的 SAML 身份提供者的 ARN。
- 对于 Condition 元素，使用 StringEquals 条件测试 SAML 响应中的 saml:aud 属性是否匹配登录控制台时浏览器显示的 URL。此登录端点 URL 是您的身份提供者的 SAML 收件人属性。您可以添加特定区域内的登录 URL。Amazon 建议使用区域端点而不是全局端点，以提高联合身份验证的韧性。有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。

  如果需要 SAML 加密，则登录 URL 必须包含 Amazon 分配给您的 SAML 提供商的唯一标识符。您可以通过在 IAM 控制台中选择身份提供者并显示详细信息页面来查看唯一标识符。

  https://*region-code*.signin.aws.amazon.com/saml/acs/*IdP-ID*

以下示例信任策略是为 SAML 联合用户设计的：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRoleWithSAML",
        "Principal": {
            "Federated": "arn:aws:iam::111122223333:saml-provider/PROVIDER-
NAME"
        },
        "Condition": {
            "StringEquals": {
                "SAML:aud": "https://region-code.signin.aws.amazon.com/saml"
            }
        }
    }
}
```

将主体 ARN 替换为您在 IAM 中创建的 SAML 提供商的实际 ARN。它会具备您自己的账户 ID 和提供商名称。

创建用于 SAML 的角色

在完成先决条件步骤后，您可以创建用于基于 SAML 的联合身份验证的角色。

要创建用于基于 SAML 的联合身份验证的角色，请执行以下操作

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，依次选择角色和创建角色。
3. 选择 SAML 2.0 联合身份验证角色类型。
4. 对于 Select a SAML provider（选择 SAML 提供商），请为您的角色选择提供商。
5. 选择 SAML 2.0 访问级别方法。

   - 选择 Allow programmatic access only (只允许编程访问) 以创建可从 Amazon API 或 Amazon CLI 以编程方式担任的角色。
   - 选择允许编程访问和 Amazon Web Services 管理控制台 访问以创建可以编程方式和从 Amazon Web Services 管理控制台 中担任的角色。

   通过这两种方法创建的角色类似，但也可从控制台担任的角色包括包含带特定条件的信任策略。该条件可以显式的方式确保将 SAML 受众（`SAML:aud` 属性）设置为 SAML 提供商的 Amazon 登录端点。
6. 定义属性的过程因访问权限类型而异。
   - 如果创建用于编程访问的角色，请从属性列表中选择一个属性。然后在 Value（值）框中，键入一个将包含在角色中的值。这样可仅限来自其 SAML 身份验证响应 (断言) 包括您指定的属性的身份提供者的用户可访问该角色。必须指定至少一个属性，以确保您的角色限于您所在组织中的一部分用户。
   - 如果您要为编程和 Amazon Web Services 管理控制台访问权限创建角色，则登录端点部分会定义在登录控制台时浏览器显示的 URL。此端点是您的身份提供者的 SAML 收件人属性，它映射到 `saml:aud` 上下文键。有关更多信息，请参阅 为身份验证响应配置 SAML 断言。。

     a. 选择区域端点或非区域端点。我们建议使用多个区域 SAML 登录端点来提高联合身份验证的韧性。

     b. 对于区域，请选择您的 SAML 提供商支持 Amazon 登录的区域。

     c. 要使登录 URL 包含唯一标识符，请选择登录端点是否包含 Amazon 分配给 SAML 身份提供者的唯一标识符。对于加密的 SAML 断言，此选项为必选。有关更多信息，请参阅 SAML 2.0 联合身份验证。

7. 要将更多与属性相关的条件添加到信任策略，请选择 Condition (optional) [条件（可选）]，选择其他条件，然后指定值。

> **Note**
>
> 列表包括最常用的 SAML 属性。IAM 支持其他可用于创建条件的属性。有关支持的属性的列表，请参阅 SAML 联合身份验证的可用键。如果需要不在列表中的支持的 SAML 属性的条件，可以手动添加此条件。为此，请在创建角色后编辑信任策略。

8. 检查 SAML 2.0 信任信息，然后选择 Next（下一步）。
9. IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限策略的策略，或者选择 Create policy（创建策略）以打开新的浏览器选项卡并从头开始创建新策略。有关更多信息，请参阅 创建 IAM 策略。在您创建策略后，关闭该选项卡并返回到您的原始选项卡。选择您希望 SAML 联合用户具有的权限策略旁边的复选框。如果您愿意，此时可以不选择任何策略，稍后将策略附加到角色。默认情况下，角色没有权限。
10. （可选）设置权限边界。这是一项高级功能。

    打开 Permissions boundary（权限边界）部分，然后选择 Use a permissions boundary to control the maximum role permissions（使用权限边界控制最大角色权限）。选择要用于权限边界的策略。
11. 选择下一步。
12. 选择下一步：审核。
13. 对于 Role name（角色名称），输入一个角色名称。角色名称在您的 Amazon Web Services 账户内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **PRODROLE** 和 **prodrole** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
14. （可选）对于 Description（描述），输入新角色的描述。
15. 在 Step 1: Select trusted entities（步骤 1：选择可信实体）或 Step 2: Add permissions（步骤 2：添加权限）部分中的 Edit（编辑），以编辑角色的用户案例和权限。
16. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
17. 检查角色，然后选择创建角色。

创建角色后，通过使用有关 Amazon 的信息来配置您的身份提供者软件，以完成 SAML 信任。此类信息包括您希望 SAML 联合用户使用的角色。这称为在 IdP 和 Amazon 之间配置信赖方信任。有关更多信息，请参阅 配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述。

使用自定义信任策略创建角色

您可以创建自定义信任策略来委派访问权限并允许其他人在您的 Amazon Web Services 账户 中执行操作。有关更多信息，请参阅 创建 IAM 策略。

有关如何使用角色委派权限的信息，请参阅角色术语和概念。

使用自定义信任策略创建 IAM 角色（控制台）

您可以使用 Amazon Web Services 管理控制台 创建 IAM 用户可担任的角色。例如，假设贵组织拥有多个 Amazon Web Services 账户 以便将开发环境与生产环境隔离。有关创建角色（该角色允许开发账户中的用户访问生产账户中的资源）的概述信息，请参阅 使用不同的开发和生产账户的示例方案。

使用自定义信任策略创建角色（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Roles，然后选择 Create role。
3. 选择 Custom trust policy（自定义信任策略）角色类型。
4. 在 Custom trust policy（自定义信任策略）部分，输入或粘贴角色的自定义信任策略。有关更多信息，请参阅 创建 IAM 策略。
5. 解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择 Next（下一步）。
6. （可选）设置权限边界。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。

   打开 Permissions boundary（权限边界）部分，然后选择 Use a permissions boundary to control the maximum role permissions（使用权限边界控制最大角色权限）。IAM 包括您的账户中的 Amazon 托管策略和客户托管策略的列表。选择要用于权限边界的策略。
7. 选择下一步。
8. 对于角色名称，角色名称自定义的程度由服务定义。如果服务定义角色的名称，则此选项不可编辑。在其他情况下，服务可能定义角色的前缀并允许您键入可选的后缀。某些服务允许您指定角色的整个名称。

   如果可能，输入角色名称或角色名称后缀。角色名称在您的 Amazon Web Services 账户 内必须是唯一的。名称不区分大小写。例如，您无法同时创建名为 **`PRODROLE`** 和 **`prodrole`** 的角色。由于其他 Amazon 资源可能引用该角色，角色创建完毕后无法编辑角色名称。
9. （可选）对于 Description（描述），输入新角色的描述。
10. （可选）在步骤 1：选择受信任的实体或步骤 2：添加权限部分中选择编辑，以编辑角色的自定义策略和权限。

11. （可选）通过以键值对的形式附加标签来向角色添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
12. 检查角色，然后选择创建角色。

委派访问权限的策略示例

以下示例演示了如何允许或授权 Amazon Web Services 账户 访问其他 Amazon Web Services 账户 中的资源。要了解如何使用这些示例 JSON 策略文档创建 IAM policy，请参阅。the section called “使用 JSON 编辑器创建策略”

主题

- 使用角色委托针对其他 Amazon Web Services 账户 资源的访问权限
- 使用策略将访问权限委托给服务
- 使用基于资源的策略委托访问另一个账户的 Amazon S3 存储桶
- 使用基于资源的策略委托针对另一个账户中的 Amazon SQS 队列的访问权限
- 当拒绝访问账户时，不得委托访问

使用角色委托针对其他 Amazon Web Services 账户 资源的访问权限

有关介绍如何使用 IAM 角色对一个账户中的用户授权以访问另一个账户中 Amazon 资源的教程，请参阅 IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限。

> **Important**
>
> 您可以在角色信任策略的 `Principal` 元素中包含特定角色或用户的 ARN。保存策略时，Amazon 将该 ARN 转换为唯一主体 ID。如果有人希望通过删除并重新创建角色或用户来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为 ARN。但是，如果您删除角色或用户，这种关系即被打破。即使您重新创建用户或角色，策略也不再适用。因为与信任策略中存储的 ID 不匹配。在这种情况下，主体 ID 会显示在控制台中，因为 Amazon 无法将其映射回 ARN。结果是，如果您删除并重新创建了信任策略的 `Principal` 元素所引用的用户或角色，您必须编辑角色，替换 ARN。当您保存策略时，它会转换为新的主体 ID。

使用策略将访问权限委托给服务

以下示例显示了一个可以附加到角色的策略。该策略可使两个服务 Amazon EMR 和 Amazon Data Pipeline 担任此角色。然后，这些服务可执行由分配给该角色 (未显示) 的权限策略授权执行的任何任务。要指定多个服务主体，不用指定两个 `Service` 元素；您可以只使用一个该元素。实际上，您可以将一组多个服务主体作为单个 `Service` 元素的值。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": [
          "elasticmapreduce.amazonaws.com",
          "datapipeline.amazonaws.com"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

使用基于资源的策略委托访问另一个账户的 Amazon S3 存储桶

在此示例中，账户 A 使用基于资源的策略（一个 Amazon S3 存储桶策略）授权账户 B 针对账户 A 的 S3 存储桶的完全访问权限。然后，账户 B 创建一个 IAM 用户策略，向账户 B 中的一个用户授予针对账户 A 的存储桶的访问权限。

账户 A 中的 S3 存储桶策略可能与以下策略类似。在此示例中，账户 A 的 S3 存储桶名为 amzn-s3-demo-bucket，账户 B 的账号为 111122223333。它在账户 B 中未指定任何单个用户或组，仅指定账户本身。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": {
    "Sid": "AccountBAccess1",
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }
}
```

或者，账户 A 可使用 Amazon S3 访问控制列表 (ACL) 来授权账户 B 访问 S3 存储桶或某个存储桶内的单个对象。在这种情况下，唯一改变的是账户 A 授权访问账户 B 的方式。如此示例的下一个部分所述，账户 B 仍使用一个策略委托针对账户 B 中的 IAM 组的访问权限。有关控制对 S3 存储桶和对象访问的更多信息，请转到 Amazon Simple Storage Service 用户指南中的访问控制。

账户 B 的管理员可能创建以下策略示例。该策略向账户 B 中的组或用户授予读取访问权限。前一策略向账户 B 授予访问权限。但是，除非组或用户策略显式授予资源访问权限，否则账户 B 中的单个组和用户不能访问资源。此策略中的权限只能是上述跨账户策略中的权限的一个子集。相比于账户 A 在第一个策略中授予账户 B 的权限，账户 B 无法向其组和用户授予更多权限。在该策略中，将显式定义 `Action` 元素以仅允许 `List` 操作，而且该策略的 `Resource` 元素与由账户 A 执行的存储桶策略的 `Resource` 匹配。

为执行该策略，账户 B 使用 IAM 将它附加到账户 B 中的相应用户（或组）。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:List*",
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket",
      "arn:aws:s3:::amzn-s3-demo-bucket/*"
    ]
  }
}
```

使用基于资源的策略委托针对另一个账户中的 Amazon SQS 队列的访问权限

在以下示例中，账户 A 有一个 Amazon SQS 队列，该队列使用附加到该队列的基于资源的策略向账户 B 授权队列访问权限。然后，账户 B 使用 IAM 组策略委托针对账户 B 中的组的访问权限。

以下示例队列策略授予账户 B 对名为 执行 queue1 的账户 A 的队列执行 `SendMessage`和 `ReceiveMessage` 操作的权限，但只在 2014 年 11 月 30 日中午至下午 3:00 之间可行。Account B 的账号为 1111-2222-3333。账户 A 使用 Amazon SQS 执行该策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Principal": {"AWS": "111122223333"},
    "Action": [
      "sqs:SendMessage",
      "sqs:ReceiveMessage"
    ],
    "Resource": ["arn:aws:sqs:*:123456789012:queue1"],
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2014-11-30T12:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2014-11-30T15:00Z"}
    }
  }
}
```

账户 B 向账户 B 中的组委托访问权限的策略可能类似于以下示例。账户 B 使用 IAM 将此策略附加到组（或用户）。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "sqs:*",
    "Resource": "arn:aws:sqs:*:123456789012:queue1"
  }
```

```
}
```

在前述 IAM 用户策略示例中，账户 B 使用通配符授权其用户访问针对账户 A 的队列的所有 Amazon SQS 操作。但是账户 B 可委托的范围仅限于账户 B 被授权访问的范围。拥有第二个策略的账户 B 组只能在 2014 年 11 月 30 日中午至下午 3:00 之间访问该队列。根据账户 A 的 Amazon SQS 队列策略的定义，用户只能执行 SendMessage 和 ReceiveMessage 操作。

当拒绝访问账户时，不得委托访问

如果其他账户已显式拒绝访问用户的父账户，则 Amazon Web Services 账户 不得委托针对该账户的资源的访问权限。此“拒绝”将传播到该账户内的所有用户，无论用户的现有策略是否授予这些用户访问权限。

例如，账户 A 编写了一个针对其账户中 S3 存储桶的存储桶策略，其中显式拒绝了账户 B 访问账户 A 的存储桶。但账户 B 编写了一个 IAM 用户策略，其中对账户 B 中的一个用户授予了对账户 A 的存储桶的访问权限。应用于账户 A 的 S3 存储桶的“显式拒绝”将传播到账户 B 中的用户。它会覆盖用于对账户 B 中用户授予访问权限的 IAM 用户策略。(有关如何计算权限的详细信息，请参阅 策略评估逻辑。)

Account A 的存储段策略可能与下列策略类似。在此示例中，账户 A 的 S3 存储桶名为 amzn-s3-demo-bucket，账户 B 的账号为 1111-2222-3333。账户 A 使用 Amazon S3 执行该策略。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "AccountBDeny",
    "Effect": "Deny",
    "Principal": {"AWS": "111122223333"},
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
  }
}
```

此“显式拒绝”将覆盖账户 B 中所有提供账户 A 中 S3 存储桶访问权限的策略。

IAM 角色管理

您必须先对用户授予切换到您创建的角色的权限，然后用户、应用程序或服务才能使用该角色。您可使用附加到组或用户的任何策略授予所需权限。本部分描述如何授予用户使用角色的权限。它还解释了用户如何从 Amazon Web Services 管理控制台、Tools for Windows PowerShell、Amazon Command Line Interface (Amazon CLI) 和 `AssumeRole` API 切换到角色。

> **Important**
>
> 当您以编程方式而不是在 IAM 控制台中创建角色，则除最长可达 64 个字符的 `RoleName` 外，您还可以选择添加最长 512 个字符的 `Path`。不过，如果您打算通过 Amazon Web Services 管理控制台 中的 Switch Role（切换角色）功能使用角色，则组合的 `Path` 和 `RoleName` 不能超过 64 个字符。

主题

- 查看角色访问
- 基于访问信息生成策略
- 向用户授予切换角色的权限
- 向用户授予权限以将角色传递给 Amazon 服务
- 撤销 IAM 角色临时安全凭证
- 更新服务相关角色
- 更新角色信任策略
- 更新角色的权限
- 更新角色的设置
- 删除角色或实例配置文件

查看角色访问

在更改角色的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

基于访问信息生成策略

有时，您可能会向 IAM 实体（用户或角色）授予超出其需要的权限。为帮助您优化授予的权限，您可以根据实体的访问活动生成 IAM policy。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含实体在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的托管策略，然后将其附加到 IAM 实体。这样，您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。要了解更多信息，请参阅 IAM Acess Analyzer 策略生成。

向用户授予切换角色的权限

当管理员创建用于跨账户存取的角色时，您在拥有角色和资源的账户（信任账户）和包含用户的账户（可信账户）之间建立了信任。为此，信任账户的管理员指定可信账号为角色的信任策略中的 `Principal`。这可能会允许可信账户中的任何用户担任该角色。要完成配置，可信账户的管理员必须为该账户中的特定组或用户提供切换到该角色的权限。

要授予切换到角色的权限

1. 作为可信账户的管理员，请为用户创建新策略，或编辑现有策略以添加所需元素。有关更多信息，请参阅 创建或编辑策略。
2. 然后，选择您希望如何分享角色信息：

   - 角色链接：向用户发送链接，以使用户进入已填写所有详细信息的 Switch Role（切换角色）页面。
   - 账户 ID 或别名：为每位用户提供角色名称以及账户 ID 号或账户别名。用户随后转到 Switch Role 页面，然后手动添加详细信息。

   有关更多信息，请参阅 向用户提供信息。

请注意，仅当您以 IAM 用户、SAML 联合角色或 web 身份联合角色登录时才能切换角色。如果您以 Amazon Web Services 账户根用户身份登录，则无法切换角色。

> **Important**
>
> 您无法将 Amazon Web Services 管理控制台中的角色切换到需要 ExternalId 值的角色。您只能通过调用支持 `ExternalId` 参数的 `AssumeRole` API 来切换到此类角色。

> **备注**
>
> - 本主题讨论了用户的策略，因为您最终会向用户授予完成任务的权限。但是，我们建议您不要向单个用户直接授予权限。当用户担任某个角色时，系统会为他们分配与该角色相关的权限。
> - 当您在 Amazon Web Services 管理控制台中切换角色时，控制台总是使用您的原始凭证对切换操作进行授权。无论您作为 IAM 用户、SAML 联合角色还是 Web 联合身份角色登录，上述情形均适用。例如，如果您切换到角色 A，则 IAM 使用您的原始用户或联合角色凭证确定是否允许您担任角色 A。如果您在使用 RoleA 时尝试切换到 RoleB，则会使用您的原始用户或联合角色凭证对您的尝试进行授权。RoleA 凭证不用于此操作。

主题

- 创建或编辑策略
- 向用户提供信息

创建或编辑策略

向用户授予担任角色权限的策略必须包括一个语句，该语句对以下项具有 `Allow` 影响：

- `sts:AssumeRole` 操作
- `Resource` 元素中该角色的 Amazon Resource Name (ARN)

获得了该策略（通过组成员资格或直接附加）的用户可以切换所列资源上的角色。

> **Note**
>
> 如果将 `Resource` 设置为 `*`，则用户可在信任用户账户的任何账户中担任何角色。（换言之，角色的信任策略将用户的账户指定为 `Principal`）。作为最佳实践，我们建议您遵循最低权限原则并仅为用户所需的角色指定完整的 ARN。

以下示例所显示的策略允许用户仅在一个账户中担任角色。此外，该策略使用通配符 (*) 来指定用户仅在角色名称以字母 `Test` 开头时才能切换到该角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "sts:AssumeRole",
        "Resource": "arn:aws:iam::111122223333:role/Test*"
    }
}
```

> **Note**
>
> 角色向用户授予的权限不会添加到用户已获得的权限。当用户切换到某个角色时，用户可临时放弃其原始权限以换取由该角色授予的权限。用户退出该角色时，将自动恢复原始用户权限。例如，假如用户的权限允许使用 Amazon EC2 实例，但是角色的权限策略未授予这些权限。在这种情况下，使用角色时，用户无法在控制台中使用 Amazon EC2 实例。此外，通过 `AssumeRole` 获取的临时凭证无法以编程方式使用 Amazon EC2 实例。

向用户提供信息

创建角色并向用户授予切换为该角色的权限后，您必须为用户提供以下信息：

- 角色的名称。
- 包含该角色的 ID 或账户别名

通过向用户发送使用账户 ID 和角色名称预配置的链接，可为用户简化访问。完成创建角色向导后，您可以选择查看角色横幅，或在任何已启用跨账户角色的角色摘要页面上查看角色链接。

您还可使用以下格式手动构建链接。请用您的账户 ID 或别名及角色名称替换以下示例中的两个参数：

`https://signin.aws.amazon.com/switchrole?account=`*`your_account_ID_or_alias`*`&roleName=`*`optional_path/role_name`*

我们建议您将用户定向到 从用户切换到 IAM 角色（控制台） 以指导他们完成该过程。要排除您在担任角色时可能遇到的常见问题，请参阅 我无法代入角色。

注意事项

- 如果您以编程方式创建角色，则可使用路径以及名称创建角色。如果执行此操作，则必须向用户提供完整的路径和角色名称，以便他们可以在 Amazon Web Services 管理控制台 的 Switch Role（切换角色）页面输入该角色。例如：`division_abc/subdivision_efg/role_XYZ`。
- 如果您以编程方式创建角色，则您可以添加最长 512 个字符的 `Path` 以及 `RoleName`。角色名称最多可以有 64 个字符。但是，要通过 Amazon Web Services 管理控制台 中的切换角色功能使用角色，则组合的 `Path` 和 `RoleName` 不能超过 64 个字符。

- 为了安全起见，您可以查看 Amazon CloudTrail 日志以了解已在 Amazon 中执行操作的人员。您可以在角色信任策略中使用 `sts:SourceIdentity` 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅 `sts:SourceIdentity`。您可以使用 `sts:RoleSessionName`，以要求用户在代入角色时指定会话名称。这可以帮助您在不同主体使用角色时区分角色会话。

向用户授予权限以将角色传递给 Amazon 服务

要配置多项 Amazon 服务，您必须将 IAM 角色传递给相应服务。这允许该服务稍后代入该角色并代表您执行操作。对于大多数服务，您只需在设置期间（而不是服务每次代入角色时）将角色传递给服务。例如，假定您的一款应用程序在 Amazon EC2 实例上运行。该应用程序需要临时凭证来进行身份验证，以及授权应用程序在 Amazon 中执行操作的权限。在设置该应用程序时，您必须将角色传递给 Amazon EC2，以便与提供这些凭证的实例一起使用。通过向角色附加 IAM policy，您为在实例上运行的应用程序定义权限。该应用程序每次需要执行角色允许的操作时都会担任该角色。

要将角色（及其权限）传递至 Amazon 服务，用户必须具有传递角色 至服务的权限。这有助于管理员确保仅批准的用户可配置具有能够授予权限的角色的服务。要允许用户将角色传递至 Amazon 服务，您必须向用户的 IAM 用户、角色或组授予 `PassRole` 权限。

> **Warning**
>
> - 只能使用 `PassRole` 权限将 IAM 角色传递给使用同一 Amazon 账户的服务。要将账户 A 中的角色传递给账户 B 中的服务，必须首先在账户 B 中创建一个可以代入账户 A 中角色的 IAM 角色，然后才能将账户 B 中的角色传递给该服务。有关详细信息，请参阅 IAM 中的跨账户资源访问。

- 请勿试图通过标记角色然后在带有 iam:PassRole 操作的策略中使用 ResourceTag 条件键来控制谁可以传递角色。这种方法没有可靠的结果。

设置 PassRole 权限时，应确保用户所传递角色的权限不会超过您希望该用户拥有的权限。例如，可能不允许 Alice 执行任何 Amazon S3 操作。如果 Alice 可以将角色传递给允许 Amazon S3 操作的服务，则该服务可以在执行作业时代表 Alice 执行 Amazon S3 操作。

在指定服务相关角色时，您还必须拥有将该角色传递给服务的权限。在某些服务中执行操作时，该服务自动在您的账户中创建一个服务相关角色。例如，在首次创建自动扩缩组时，Amazon EC2 Auto Scaling 会为您创建 AWSServiceRoleForAutoScaling 服务相关角色。如果您在创建自动扩缩组时尝试指定服务相关角色，但您没有 iam:PassRole 权限，则会收到错误。如果您没有明确指定角色，则不需要 iam:PassRole 权限，默认情况下，对该组执行的所有操作都使用 AWSServiceRoleForAutoScaling 角色。要了解哪些服务支持服务相关角色，请参阅使用 IAM 的 Amazon 服务。要了解在哪些服务中执行操作时自动创建服务相关角色，请选择是链接并查看该服务的服务相关角色文档。

在任何使用角色分配服务权限的 API 操作中，用户都可以将角色 ARN 作为参数进行传递。该服务随后检查该用户是否拥有 iam:PassRole 权限。要限制用户只传递批准的角色，您可以使用 IAM policy 语句的 iam:PassRole 元素筛选 Resources 权限。

您可以在 JSON 策略中使用 Condition 元素来测试所有 Amazon 请求的请求上下文中所包含键的值。要了解有关在策略中使用条件键的更多信息，请参阅 IAM JSON 策略元素：Condition。iam:PassedToService 条件键可用于指定可将角色传递到的服务的服务主体。要了解有关在策略中使用 iam:PassedToService 条件键的更多信息，请参阅 iam:PassedToService。

示例 1

假设您要授予用户在启动实例时能够将任意批准角色组传递至 Amazon EC2 服务的能力。您需要三个部分：

- IAM 权限策略附加到确定角色可执行哪些任务的角色。将权限范围限定为仅角色必须执行的操作，以及角色进行这些操作所需的资源。您可以使用 Amazon 托管的或客户创建的 IAM 权限策略。

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
```

```
        "Effect": "Allow",
        "Action": [ "A list of the permissions the role is allowed to use" ],
        "Resource": [ "A list of the resources the role is allowed to access" ]
    }
}
```

- 允许服务担任角色的信任策略。例如，您可以将以下信任策略与具有 UpdateAssumeRolePolicy 操作的角色进行附加。该信任策略允许 Amazon EC2 使用角色和附加在角色上的权限。

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid":
 "TrustPolicyStatementThatAllowsEC2ServiceToAssumeTheAttachedRole",
        "Effect": "Allow",
        "Principal": { "Service": "ec2.amazonaws.com" },
       "Action": "sts:AssumeRole"
    }
}
```

- 附加至 IAM 用户的 IAM permissions policy（权限策略），允许该用户仅传递已获批准的那些策略。您通常将 iam:GetRole 添加至 iam:PassRole，使用户能够获取准备进行传递的角色的详细信息。在此示例中，用户只能传递位于指定账户中并且名称以 EC2-roles-for-XYZ- 开头的角色：

  JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:PassRole"
            ],
            "Resource": "arn:aws:iam::111122223333:role/EC2-roles-for-XYZ-*"
        }
    ]
}
```

现在用户可以启动具有所分配角色的 Amazon EC2 实例。实例上运行的应用程序可以通过实例配置文件元数据访问角色的临时许可。附加到角色的许可策略确定实例可以执行的任务。

示例 2

Amazon Relational Database Service (Amazon RDS) 支持名为 Enhanced Monitoring（增强监控）的功能。此功能使 Amazon RDS 能够使用代理监控数据库实例。它还允许 Amazon RDS 将指标记录到 Amazon CloudWatch Logs 中。要启用此功能，您必须创建一个服务角色，以便为 Amazon RDS 提供监控指标和将指标写入日志的权限。

为 Amazon RDS 增强监控创建角色

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择 角色，然后选择 创建角色。
3. 选择 Amazon 服务角色类型，然后在适用于其他 Amazon Web Services 服务 的使用案例中，选择 RDS 服务。选择 RDS - Enhanced Monitoring（RDS - 增强监控），然后选择 Next（下一步）。
4. 选择 AmazonRDSEnhancedMonitoringRole 权限策略。
5. 选择下一步。
6. 对于 Role name（角色名称），键入有助于标识此角色作用的角色名称。角色名称在您的 Amazon Web Services 账户内必须是唯一的。在策略中使用角色名称或将其作为 ARN 的一部分时，角色名称区分大小写。在控制台中向客户显示角色名称时（例如在登录过程中），角色名称不区分大小写。由于多个实体可能引用该角色，因此，角色创建完毕后，您将无法编辑角色名称。
7. （可选）对于 Description（描述），输入新角色的描述。
8. （可选）通过以键值对的形式附加标签来向用户添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
9. 检查角色，然后选择创建角色。

角色自动获得授予 `monitoring.rds.amazonaws.com` 服务担任角色权限的信任策略。在此之后，Amazon RDS 可执行 `AmazonRDSEnhancedMonitoringRole` 策略允许的所有操作。

您希望使用增强监控的用户需要包括允许用户列出 RDS 角色的语句和允许用户传递角色的语句的策略，如下所示。使用您的账号，然后使用您在步骤 6 中提供的名称替换角色名称。

```
    {
```

```
      "Sid": "PolicyStatementToAllowUserToListRoles",
      "Effect": "Allow",
      "Action": ["iam:ListRoles"],
      "Resource": "*"
    },
    {
        "Sid": "PolicyStatementToAllowUserToPassOneSpecificRole",
        "Effect": "Allow",
        "Action": [ "iam:PassRole" ],
        "Resource": "arn:aws:iam::account-id:role/RDS-Monitoring-Role"
    }
```

您可以将该语句与另一策略中的语句进行合并，或将此语句放在其自身策略中。如要指定用户可传递以 RDS- 开始的任何角色，您可以在资源 ARN 中使用通配符替换角色名称，如下所示。

```
        "Resource": "arn:aws:iam::account-id:role/RDS-*"
```

Amazon CloudTrail 日志中的 **iam:PassRole** 操作

PassRole 不是 API 调用。PassRole 是一种权限，意味着不会为 IAM PassRole 生成 CloudTrail 日志。要查看向 CloudTrail 中的哪个 Amazon Web Services 服务 传递了哪些角色，您必须查看创建或修改接收相应角色的 Amazon 资源的 CloudTrail 日志。例如，角色在创建时会传递给 Amazon Lambda 函数。CreateFunction 操作的日志显示了传递给该函数的角色记录。

撤销 IAM 角色临时安全凭证

> **Warning**
>
> 如果您执行此页面上的步骤，则通过担任角色创建的具有当前会话的所有用户对所有 Amazon 操作和资源的访问将被拒绝。这会导致用户丢失未保存的工作。

在允许用户访问具有较长的会话持续时间（例如 12 小时）的 Amazon Web Services 管理控制台 时，用户的临时凭证不会很快过期。如果用户无意中向未授权第三方公开其凭证，则第三方在会话的持续时间内将具有访问权限。不过，如果需要，可以撤销对某个特定时间点之前发布的角色凭证的所有权限。指定时间之前发布的该角色的所有临时凭证将变得无效。这将强制所有用户重新进行身份验证并请求新的凭证。

> **Note**
>
> 您无法撤销服务相关角色对会话的权限。

在使用本主题中的过程撤消角色的权限时，Amazon 会向角色附加新的内联策略来拒绝对所有操作的所有权限。它包括仅当用户在撤销权限时的某个时间点之前 代入角色的情况下应用限制的情况。如果用户在您撤销权限之后 代入角色，则拒绝策略不适用于该用户。

有关拒绝访问的更多信息，请参阅 禁用临时安全凭证的权限。

> **Important**
>
> 此拒绝策略适用于指定角色的所有用户，而不只是适用于具有持续时间更长的控制台会话的用户。

从角色撤消会话权限所需的最低权限

要从角色成功撤消会话权限，您必须具有该角色的 `PutRolePolicy` 权限。这允许您将 `AWSRevokeOlderSessions` 内联策略附加到该角色。

撤消会话权限

您可以撤消某个角色的会话权限，以拒绝代入该角色的任何用户的所有权限。

> **Note**
>
> 您不能编辑 IAM 中根据 IAM Identity Center 权限集创建的角色。您必须在 IAM Identity Center 中撤消用户的活动权限集会话。有关更多信息，请参阅《IAM Identity Center 用户指南》中的撤消由权限集创建的活动 IAM 角色会话。

立即拒绝对角色凭证的任何当前用户的所有权限

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选择要撤消其权限的角色的名称（而不是复选框）。
3. 在所选角色的 Summary 页面上，选择 Revoke sessions 选项卡。

4. 在 Revoke sessions 选项卡上，选择 Revoke active sessions。
5. Amazon 要求您确认此操作。选择我确认我正在撤消此角色的所有活动会话。复选框，然后在对话框中选择撤消活动会话。

   然后 IAM 会将名为 `AWSRevokeOlderSessions` 的策略附加到角色。选择撤消活动会话后，此策略将拒绝在过去已经以及未来大约 30 秒钟将会代入该角色的用户的所有访问权限。此未来时间选项考虑了策略的传播延迟，以便处理在更新后的策略在给定区域生效之前获得或续订的新会话。任何在您选择“撤消活动会话”之后超过大约 30 秒代入角色的用户均不受影响。要了解为什么更改并非始终立即可见，请参阅 我所做的更改可能不会立即可见。

> **Note**
>
> 如果您稍后再次选择撤消活动会话，则将刷新策略中的日期和时间戳，并将再次拒绝在新的指定时间之前代入角色的任何用户的所有访问权限。

以这种方式调用会话的有效用户必须获得临时凭证，新会话才能继续工作。在凭证过期前，Amazon CLI 会缓存凭证。要强制 CLI 删除并刷新已失效的缓存证书，请运行以下命令之一：

Linux、macOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

在指定时间之前撤消会话权限

您也可以使用 Amazon CLI 或 SDK 在策略的 Condition 元素中指定 `aws:TokenIssueTime` 键的值，随时撤消会话权限。

当 `aws:TokenIssueTime` 的值早于指定的日期和时间时，该策略会拒绝所有权限。`aws:TokenIssueTime` 的值对应于临时安全凭证的确切创建时间。`aws:TokenIssueTime` 值仅存在于使用临时安全凭证签署的 Amazon 请求的上下文中，因此，该策略中的 Deny 语句不会影响使用该 IAM 用户的长期凭证签署的请求。

该策略还可以附加到角色。在这种情况下，该策略只会影响由该角色在指定日期和时间之前创建的临时安全凭证。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "*",
    "Resource": "*",
    "Condition": {
      "DateLessThan": {"aws:TokenIssueTime": "2014-05-07T23:47:00Z"}
    }
  }
}
```

以这种方式调用会话的有效用户必须获得临时凭证，新会话才能继续工作。在凭证过期前，Amazon CLI 会缓存凭证。要强制 CLI 删除并刷新已失效的缓存证书，请运行以下命令之一：

Linux、macOS 或 Unix

```
$ rm -r ~/.aws/cli/cache
```

Windows

```
C:\> del /s /q %UserProfile%\.aws\cli\cache
```

更新服务相关角色

您用来编辑服务相关角色的方法取决于服务。某些服务可能允许您从服务控制台、API 或 CLI 编辑服务相关角色的权限。但是，创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。您可以从 IAM 控制台、API 或 CLI 编辑任何角色的描述。

有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。要了解服务是否支持编辑服务相关角色，请选择 Yes 链接以查看该服务的服务相关角色文档。

编辑服务相关角色描述（控制台）

您可以使用 IAM 控制台编辑服务相关角色的描述。

编辑服务相关角色的描述（控制台）

1. 在 IAM 控制台的导航窗格中，选择角色。
2. 以下代码示例显示如何将 IAM 策略附加到用户。
3. 在 Role description 的最右侧，选择 Edit。
4. 在框中输入新描述，然后选择 Save（保存）。

编辑服务相关角色描述 (Amazon CLI)

您可以从 Amazon CLI 使用 IAM 命令编辑服务相关角色的描述。

更改服务相关角色描述 (Amazon CLI)

1. （可选）要查看角色的当前描述，请运行以下命令：

   ```
   aws iam get-role --role-name ROLE-NAME
   ```

   通过 CLI 命令使用角色名称（并非 ARN）指向角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **myrole**。
2. 要更新服务相关角色的描述，请运行以下命令：

   ```
   aws iam update-role --role-name ROLE-NAME --description OPTIONAL-DESCRIPTION
   ```

编辑服务相关角色描述 (Amazon API)

您可以使用 Amazon API 编辑服务相关角色的描述。

更改服务相关角色的描述 (Amazon API)

1. （可选）要查看角色的当前描述，请调用以下操作，并指定角色的名称：

   Amazon API：GetRole
2. 要更新角色的描述，请调用以下操作，并指定角色的名称（和可选描述）：

   Amazon API：UpdateRole

更新角色信任策略

要更改可担任角色的人员，您必须修改角色的信任策略。您无法修改服务相关角色的信任策略。

> **备注**
>
> - 如果用户被列为角色的信任策略中的主体，但无法担任该角色，请检查用户的权限边界。如果为用户设置了权限边界，则它必须允许该 `sts:AssumeRole` 操作。
> - 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源（例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda）的 Amazon Web Services 服务会提供临时凭证并自动更新这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 会话标签 或者 会话策略，则需要重新代入当前角色。

更新角色信任策略（控制台）

更改 Amazon Web Services 管理控制台 中的角色信任策略

1. 登录 Amazon Web Services 管理控制台，打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 在您的账户的角色列表中，选择要修改的角色的名称。
4. 选择 Trust relationships（信任关系）选项卡，然后选择 Edit trust policy（编辑信任策略）。
5. 根据需要编辑信任策略。要添加其他可担任角色的主体，请在 `Principal` 元素中指定他们。例如，以下策略代码段演示如何在 `Principal` 元素中引用两个 Amazon Web Services 账户：

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:root",
    "arn:aws:iam::444455556666:root"
  ]
},
```

如果您指定其他账户中的主体，将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下，受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担任角色的权限。为此，管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息，请参阅以下过程或向用户授予切换角色的权限。

以下策略代码段演示如何在 `Principal` 元素中引用两个 Amazon 服务：

```
"Principal": {
  "Service": [
    "opsworks.amazonaws.com",
    "ec2.amazonaws.com"
  ]
},
```

6. 在编辑完信任策略后，请选择 Update policy（更新策略）以保存所做更改。

   有关策略结构和语法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限和IAM JSON 策略元素参考。

允许可信外部账户中的用户使用角色（控制台）

有关此过程的更多详细信息，请参阅向用户授予切换角色的权限。

1. 登录受信任外部 Amazon Web Services 账户。
2. 确定将权限附加到用户还是附加到组。在 IAM 控制台的导航窗格中，相应选择 Users（用户）或 User Groups（用户组）。
3. 选择您要向其授予访问权限的用户或组的名称，然后选择 Permissions 选项卡。
4. 请执行以下操作之一：

   - 要编辑某个客户 托管策略，请选择该策略的名称，选择编辑策略，然后选择 JSON 选项卡。您不能编辑 Amazon 托管策略。Amazon 托管策略随 Amazon 图标 ( ) 一起显示。有关 Amazon 托管策略与客户托管策略之间的差别的更多信息，请参阅托管策略与内联策略。
   - 要编辑某个内联策略，请选择该策略名称旁边的箭头，然后选择 Edit policy。

5. 在策略编辑器中，添加一个新的 `Statement` 元素，指定以下内容：

```
{
  "Effect": "Allow",
  "Action": "sts:AssumeRole",
  "Resource": "arn:aws:iam::ACCOUNT-ID:role/ROLE-NAME"
}
```

将语句中的 ARN 替换为用户可担任的角色的 ARN。

6. 按照屏幕上的提示完成策略的编辑。

更新角色信任策略（Amazon CLI）

您可以使用 Amazon CLI 更改可担任角色的人员。

修改角色信任策略 (Amazon CLI)

1. （可选）如果不知道要修改的角色的名称，请运行以下命令以列出账户中的角色：

   - aws iam list-roles

2. （可选）要查看角色当前的信任策略，请运行以下命令：

   - aws iam get-role

3. 要修改可访问角色的受信任主体，请创建带有已更新信任策略的文本文件。您可以使用任何文本编辑器构件策略。

   例如，以下信任策略说明了如何在 `Principal` 元素中引用两个 Amazon Web Services 账户。这允许两个单独的 Amazon Web Services 账户 中的用户代入此角色。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"AWS": [
            "arn:aws:iam::111122223333:root",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "sts:AssumeRole"
    }
```

```
}
```

如果您指定其他账户中的主体，将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下，受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担任角色的权限。为此，管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息，请参阅以下过程或向用户授予切换角色的权限。

4. 要使用您刚刚创建的文件来更新信任策略，请运行以下命令：

   - aws iam update-assume-role-policy

允许受信任外部账户中的用户使用角色 (Amazon CLI)

有关此过程的更多详细信息，请参阅向用户授予切换角色的权限。

1. 创建一个 JSON 文件，其中包含授予担任角色的权限的权限策略。例如，下面的策略包含最低必需权限：

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Action": "sts:AssumeRole",
           "Resource": "arn:aws:iam::111122223333:role/ROLE-NAME"
       }
   }
   ```

   将语句中的 ARN 替换为用户可担任的角色的 ARN。

2. 运行以下命令来将包含信任策略的 JSON 文件上传到 IAM：

   - aws iam create-policy

   此命令的输出包含策略的 ARN。请记录此 ARN，因为您在后面的步骤中需要它。

3. 决定要将策略附加到哪个用户或组。如果不知道目标用户或组的名称，请使用下列命令之一列出账户中的用户或组：

- aws iam list-users
- aws iam list-groups

4. 使用以下命令之一，将您在上一步中创建的策略附加到用户或组：

   - aws iam attach-user-policy
   - aws iam attach-group-policy

更新角色信任策略（Amazon API）

您可以使用 Amazon API 更改可担任角色的人员。

修改角色信任策略 (Amazon API)

1. （可选）如果不知道要修改的角色的名称，请调用以下操作以列出账户中的角色：

   - ListRoles

2. （可选）要查看角色当前的信任策略，请调用以下操作：

   - GetRole

3. 要修改可访问角色的受信任主体，请创建带有已更新信任策略的文本文件。您可以使用任何文本编辑器构件策略。

   例如，以下信任策略说明了如何在 `Principal` 元素中引用两个 Amazon Web Services 账户。这允许两个单独的 Amazon Web Services 账户 中的用户代入此角色。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Principal": {"AWS": [
            "arn:aws:iam::111122223333:root",
            "arn:aws:iam::444455556666:root"
        ]},
        "Action": "sts:AssumeRole"
    }
}
```

如果您指定其他账户中的主体，将账户添加到角色的信任策略只是建立跨账户信任关系工作的一半而已。默认情况下，受信任账户中的任何用户均无法担任角色。新的受信任账户的管理员必须授予用户担任角色的权限。为此，管理员必须创建或编辑附加到用户以允许该用户访问 `sts:AssumeRole` 操作的策略。有关更多信息，请参阅以下过程或向用户授予切换角色的权限。

4. 要使用您刚刚创建的文件来更新信任策略，请调用以下操作：

   - UpdateAssumeRolePolicy

允许可信外部账户中的用户使用角色 (Amazon API)

有关此过程的更多详细信息，请参阅向用户授予切换角色的权限。

1. 创建一个 JSON 文件，其中包含授予担任角色的权限的权限策略。例如，下面的策略包含最低必需权限：

   JSON

   ```
   {
       "Version":"2012-10-17",
       "Statement": {
           "Effect": "Allow",
           "Action": "sts:AssumeRole",
           "Resource": "arn:aws:iam::111122223333:role/ROLE-NAME"
       }
   }
   ```

   将语句中的 ARN 替换为用户可担任的角色的 ARN。

2. 调用以下操作来将包含信任策略的 JSON 文件上传到 IAM：

   - CreatePolicy

   此操作的输出包含策略的 ARN。请记录此 ARN，因为您在后面的步骤中需要它。

3. 决定要将策略附加到哪个用户或组。如果不知道目标用户或组的名称，请调用下列操作之一列出账户中的用户或组：

   - ListUsers

- ListGroups

4. 调用以下操作之一，将您在上一步中创建的策略附加到用户或组：

   - API：AttachUserPolicy
   - AttachGroupPolicy

更新角色的权限

使用以下过程更新角色的权限策略和权限边界。

先决条件：查看角色访问权限

在更改角色的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

更新角色的权限策略

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的服务相关角色的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色中列为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

更新角色权限策略（控制台）

更改角色允许的权限 (控制台)

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 选择要修改的角色的名称，然后选择权限选项卡。
4. 请执行以下操作之一：

   - 要编辑某个现有客户托管策略，请选择该策略的名称，然后选择 Edit policy。

   > **Note**
   >
   > 您不能编辑 Amazon 托管策略。Amazon 托管策略随 Amazon 图标
   > ( )

一起显示。有关 Amazon 托管策略与客户托管策略之间的差别的更多信息，请参阅托管策略与内联策略。

- 要将现有的托管策略附加到角色，请选择 Add permissions（添加权限），然后选择 Attach policies（附加策略）。
- 要编辑现有内联策略，请展开策略并选择 Edit（编辑）。
- 要嵌入新的内联策略，请选择 Add permissions（添加权限），然后选择 Create inline policy（创建内联策略）。
- 若要从角色中移除现有策略，请选中策略名称旁边的复选框，然后选择删除。

更新角色权限策略（Amazon CLI）

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的服务相关角色的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色中列为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

更改角色允许的权限 (Amazon CLI)

1. （可选）如需查看当前与角色关联的权限，请运行以下命令：
   1. aws iam list-role-policies（用于列出内联策略）
   2. aws iam list-attached-role-policies（用于列出托管策略）
2. 对于更新角色的权限所使用的命令，根据您是在更新托管策略还是内联策略而有所不同。

   要更新托管策略，请运行以下命令以创建托管策略的新版本：

   - aws iam create-policy-version

   要更新内联策略，请运行以下命令：

   - aws iam put-role-policy

更新角色权限策略（Amazon API）

要更改该角色允许的权限，请修改该角色的权限策略。您无法修改 IAM 中的服务相关角色的权限策略。您可能能够修改依赖角色的服务中的权限策略。要检查服务是否支持此功能，请参阅使用 IAM 的

Amazon 服务并查找服务相关角色中列为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

更改角色允许的权限 (Amazon API)

1. （可选）如需查看当前与角色关联的权限，请调用以下操作：
    1. ListRolePolicies（用于列出内联策略）
    2. ListAttachedRolePolicies（用于列出托管策略）
2. 对于更新角色的权限所使用的操作，根据您是在更新托管策略还是内联策略而有所不同。

    要更新托管策略，请调用以下操作以创建托管策略的新版本：

    - CreatePolicyVersion

    要更新内联策略，请调用以下操作：

    - PutRolePolicy

更新角色的权限边界

要更改对某角色允许的最大权限，请修改角色的权限边界。

更新角色权限边界（控制台）

更改用于设置角色的权限边界的策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 选择具有您要更改的 权限边界 的角色名称。
4. 选择权限选项卡。如有必要，打开 Permissions boundary（权限边界）部分，然后选择 Change boundary（更改边界）。
5. 选择要用于权限边界的策略。
6. 选择 Change boundary（更改边界）。

    在某个人下次担任该角色后，您所做的更改才会生效。

更新角色权限边界（Amazon CLI）

更改用于设置角色的权限边界的托管策略 (Amazon CLI)

1. （可选）要查看角色的当前权限边界，请运行以下命令：
   - aws iam get-role
2. 要使用不同的托管策略来更新角色的权限边界，请运行以下命令：
   - aws iam put-role-permissions-boundary

   角色只能具有一个设置为权限边界的托管策略。如果您更改权限边界，则会更改允许的角色的最大权限。

更新角色权限边界（Amazon API）

更改用于设置角色的权限边界的托管策略 (Amazon API)

1. （可选）要查看角色的当前权限边界，请调用以下操作：
   - GetRole
2. 要使用不同的托管策略来更新角色的权限边界，请调用以下操作：
   - PutRolePermissionsBoundary

   角色只能具有一个设置为权限边界的托管策略。如果您更改权限边界，则会更改允许的角色的最大权限。

更新角色的设置

使用以下过程更新角色的描述或更改角色的最长会话持续时间。

更新角色描述

要更改角色的描述，请修改描述文本。

更新角色描述（控制台）

更改角色的描述 (控制台)

1. 登录 Amazon Web Services 管理控制台，打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 以下代码示例显示如何将 IAM 策略附加到用户。
4. 在 Summary（摘要）部分中，选择 Edit（编辑）。
5. 在框中键入新描述，然后选择 Save changes (保存更改)。

更新角色描述（Amazon CLI）

更改角色的描述 (Amazon CLI)

1. （可选）要查看角色的当前描述，请运行以下命令：
   - aws iam get-role
2. 要更新角色的描述，请带描述参数运行以下命令：
   - aws iam update-role

更新角色描述（Amazon API）

更改角色的描述 (Amazon API)

1. （可选）要查看角色当前的描述，请调用以下操作：
   - GetRole
2. 要更新角色的描述，请带描述参数调用以下操作：
   - UpdateRole

更新角色的最长会话持续时间

要为使用控制台、Amazon CLI 或 Amazon API 代入的角色指定最大会话持续时间设置，请修改最大会话持续时间设置值。该设置可以具有 1 小时到 12 小时之间的值。如果未指定值，则应用默认最大值 (1 小时)。该设置不限制 Amazon 服务建立的会话。

更新最长角色会话持续时间（控制台）

更改使用控制台、Amazon CLI 或 Amazon API 担任的角色的最大会话持续时间设置（控制台）

1. 登录 Amazon Web Services 管理控制台，打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色。
3. 以下代码示例显示如何将 IAM 策略附加到用户。
4. 在 Summary（摘要）部分中，选择 Edit（编辑）。
5. 对于 Maximum session duration（最大会话持续时间），请选择一个值。您还可以选择 Custom duration（自定义持续时间）并输入一个值（以秒为单位）。
6. 选择保存更改。

   在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅撤销 IAM 角色临时安全凭证。

在预设情况下，在 Amazon Web Services 管理控制台 中的 IAM 用户会话为 12 小时。在控制台内切换角色的 IAM 用户被授予角色最大会话持续时间或用户会话中的剩余时间（以较少者为准）。

从 Amazon CLI 或 Amazon API 代入角色的任何人都可以请求更长的会话，最多达到这个最大值。`MaxSessionDuration` 设置确定可请求的最大角色会话的持续时间。

- 要使用 Amazon CLI 指定会话持续时间，请使用 `duration-seconds` 参数。要了解更多信息，请参阅切换到 IAM 角色（Amazon CLI）。
- 要使用 Amazon API 指定会话持续时间，请使用 `DurationSeconds` 参数。要了解更多信息，请参阅切换到 IAM 角色（Amazon API）。

更新最长角色会话持续时间（Amazon CLI）

> **Note**
>
> 从 Amazon CLI 或 API 中担任角色的任何人都可以使用 `duration-seconds` CLI 参数或 `DurationSeconds` API 参数请求更长的会话。`MaxSessionDuration` 设置确定可使用 `DurationSeconds` 参数请求的最大角色会话持续时间。如果用户未指定 `DurationSeconds` 参数值，其安全凭证的有效期为 1 小时。

使用 Amazon CLI (Amazon CLI) 更改担任的角色的最大会话持续时间设置

1. （可选）要查看角色的当前最大会话持续时间设置，请运行以下命令：

   - aws iam get-role

2. 要更新角色的最大会话持续时间设置，请带 `max-session-duration` CLI 参数或 `MaxSessionDuration` API 参数运行以下命令：

   - aws iam update-role

   在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅撤销 IAM 角色临时安全凭证。

更新最长角色会话持续时间（Amazon API）

> **Note**
>
> 从 Amazon CLI 或 API 中担任角色的任何人都可以使用 `duration-seconds` CLI 参数或 `DurationSeconds` API 参数请求更长的会话。`MaxSessionDuration` 设置确定可使用 `DurationSeconds` 参数请求的最大角色会话持续时间。如果用户未指定 `DurationSeconds` 参数值，其安全凭证的有效期为 1 小时。

使用 API 更改担任的角色的最大会话持续时间设置 (Amazon API)

1. （可选）要查看角色的当前最大会话持续时间设置，请调用以下操作：

   - GetRole

2. 要更新角色的最大会话持续时间设置，请带 `max-session-duration` CLI 参数或 `MaxSessionDuration` API 参数调用以下操作：

   - UpdateRole

   在某个人下次担任该角色后，您所做的更改才会生效。要了解如何撤销该角色的现有会话，请参阅撤销 IAM 角色临时安全凭证。

删除角色或实例配置文件

如果您不再需要某个角色，我们建议您删除该角色及其关联的权限。这样您就没有未被主动监控或维护的未使用实体。

如果该角色与 EC2 实例关联，您还可以从实例配置文件中删除角色，然后删除实例配置文件。

> **Warning**
>
> 确保您没有使用要删除的角色或实例配置文件运行任何 Amazon EC2 实例。删除与正在运行的实例关联的角色或实例配置文件将中断该实例上正在运行的所有应用程序。

如果您不希望永久删除角色，则可以禁用角色。为此，请更改角色的策略，然后撤消所有当前会话。例如，您可以将策略添加到拒绝访问所有 Amazon 的角色。您还可以编辑信任策略，以拒绝任何试图代入此角色的用户的访问。有关撤消会话的更多信息，请参阅 撤销 IAM 角色临时安全凭证。

主题

- 查看角色访问权限
- 删除服务相关角色
- 删除 IAM 角色（控制台）
- 创建 IAM 角色 (Amazon CLI)
- 删除 IAM 角色 (Amazon API)
- 相关信息

查看角色访问权限

在删除角色之前，建议您查看上次使用角色的时间。可使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 完成此操作。如果您不希望从使用该角色的用户处删除访问权限，则应查看此信息。

角色的上次活动日期可能与上次访问时间选项卡中报告的上次日期不相符。上次访问时间选项卡仅报告角色的权限策略所允许服务的活动。角色的上次活动日期包括访问最后一次尝试 Amazon 中的任何服务。

> **Note**
>
> 角色上次活动和上次访问数据的跟踪周期为 400 天。如果您的地区在一年内开始支持这些功能，则此时间段可能会缩短。角色可能在 400 天之前使用过。有关跟踪周期的更多信息，请参阅Amazon 跟踪上次访问信息的位置。

查看上次使用角色的时间（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 查找要查看其活动的角色的行。您可以使用搜索字段缩小结果范围。查看上一个活动列以查看距离上次使用角色的天数。如果在跟踪周期内未使用该角色，则表将显示无。
4. 选择角色的名称可查看更多信息。角色的摘要页面还包括上一个活动，显示上次使用角色的日期。如果在过去 400 天内未使用该角色，则上一个活动中显示在跟踪期间未被访问。

查看上次使用角色的时间 (Amazon CLI)

`aws iam get-role` – 运行此命令返回有关角色的信息，包括 `RoleLastUsed` 对象。此对象包含上次使用角色的 `Region` 和 `LastUsedDate`。如果存在 `RoleLastUsed` 但不包含值，则在跟踪周期内未使用该角色。

查看上次使用角色的时间 (Amazon API)

`GetRole` – 调用此操作以返回有关角色的信息，包括 `RoleLastUsed` 对象。此对象包含上次使用角色的 `Region` 和 `LastUsedDate`。如果存在 `RoleLastUsed` 但不包含值，则在跟踪周期内未使用该角色。

删除服务相关角色

您用来删除服务相关角色的方法取决于服务。在某些情况下，无需手动删除服务相关角色。例如，在服务中完成特定操作 (如删除资源) 时，服务可能为您删除服务相关角色。在其他情况下，服务可能支持从服务控制台、API 或 Amazon CLI 中手动删除服务相关角色。

查看链接服务中服务相关角色的文档以了解如何删除该角色。您可以转至控制台中的 IAM Roles（角色）页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities（可信实体）列中随 (Service-linked role) [（服务相关角色）] 一起显示。角色的 Summary（摘要）页面上的横幅也指示角色是服务相关角色。

如果服务不包括有关删除服务相关角色的文档，您可以使用 IAM 控制台、Amazon CLI 或 API 来删除角色。

删除 IAM 角色（控制台）

当您使用 Amazon Web Services 管理控制台 删除角色时，IAM 会自动分离与该角色关联的托管式策略。此外，它还会自动删除任何与该角色关联的内联策略，以及包含该角色的任何 Amazon EC2 实例配置文件。

> **Important**
>
> 在某些情况下，某个角色可能与 Amazon EC2 实例配置文件关联，并且该角色和实例配置文件可能同名。在这种情况下，您可以使用 Amazon Web Services 管理控制台删除角色和实例配置文件。对于您在控制台中创建的角色和实例配置文件，会自动建立这种关联。如果您从 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 创建角色，则角色和实例配置文件可能具有不同的名称。在这种情况下，您无法使用控制台删除它们。而是必须使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 首先从实例配置文件中删除该角色。然后必须采取单独的步骤删除该角色。

删除角色 (控制台)

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色，然后选中要删除的角色旁的复选框。
3. 在页面的顶部，选择 Delete（移除）。
4. 在确认对话框中，查看上次访问的信息，该信息显示每个选定角色上次访问 Amazon 服务的时间。这样可帮助您确认角色当前是否处于活动状态。如果要继续操作，请在文本输入字段中输入角色的名称，然后选择 Delete（删除）。如果确定，您就可以继续执行删除操作，即使仍在加载上次访问的信息。

> **Note**
>
> 您不能使用控制台删除实例配置文件，除非它与角色同名。实例配置文件在删除角色的过程中被删除，如上述步骤所述。要在不删除角色的情况下删除实例配置文件，必须使用 Amazon CLI 或 Amazon API。有关更多信息，请参阅以下部分。

创建 IAM 角色 (Amazon CLI)

当您使用 Amazon CLI 删除角色时，必须先删除与角色关联的内联策略。您还必须分离与该角色关联的托管式策略。如果您想要删除包含角色的关联实例配置文件，必须分别删除。

删除角色 (Amazon CLI)

1. 如果您不知道要删除的角色的名称，请输入以下命令列出账户中的角色：

   ```
   aws iam list-roles
   ```

   该列表包括各个角色的 Amazon Resource Name (ARN)。通过 CLI 命令使用角色名称（并非 ARN）指向角色。例如，如果某个角色的 ARN 为 `arn:aws:iam::123456789012:role/myrole`，则将该角色称为 **`myrole`**。

2. 从该角色所关联的全部实例配置文件中删除该角色。

   a. 要列出与角色关联的所有实例配置文件，请输入以下命令：

      ```
      aws iam list-instance-profiles-for-role --role-name role-name
      ```

   b. 要从某个实例配置文件中删除该角色，请为每个实例配置文件输入以下命令：

      ```
      aws iam remove-role-from-instance-profile --instance-profile-name instance-profile-name --role-name role-name
      ```

3. 删除与角色相关的全部策略。

   a. 要列出角色中的所有内联策略，请输入以下命令：

      ```
      aws iam list-role-policies --role-name role-name
      ```

   b. 要删除角色中的每个内联策略，请为每个策略输入以下命令：

      ```
      aws iam delete-role-policy --role-name role-name --policy-name policy-name
      ```

   c. 要列出附加到角色的所有托管策略，请输入以下命令：

      ```
      aws iam list-attached-role-policies --role-name role-name
      ```

   d. 要从角色中分离每个托管策略，请为每个策略输入以下命令：

```
aws iam detach-role-policy --role-name role-name --policy-arn policy-arn
```

4. 输入以下命令可删除角色：

```
aws iam delete-role --role-name role-name
```

5. 如果您没有计划重新使用与角色相关联的实例配置文件，则可以输入以下命令来删除它们：

```
aws iam delete-instance-profile --instance-profile-name instance-profile-name
```

删除 IAM 角色 (Amazon API)

当您使用 IAM API 删除角色时，必须先删除与角色关联的内联策略。您还必须分离与该角色关联的托管式策略。如果您想要删除包含角色的关联实例配置文件，必须分别删除。

删除角色 (Amazon API)

1. 要列出角色所关联的所有实例配置文件，请调用 ListInstanceProfilesForRole。

   要从实例配置文件删除该角色，请调用 RemoveRoleFromInstanceProfile。必须提交角色名称和实例配置文件名称。

   如果您不会重用与角色相关联的实例配置文件，请调用 DeleteInstanceProfile 来删除它。
2. 要列出一个角色的所有内联策略，请调用 ListRolePolicies。

   要删除该角色所关联的所有内联策略，请调用 DeleteRolePolicy。必须提交角色名称和内联策略名称。
3. 要列出附加到角色的所有托管式策略，请调用 ListAttachedRolePolicies。

   要分离附加到角色的托管式策略，请调用 DetachRolePolicy。必须提交角色名称和托管策略 ARN。
4. 调用 DeleteRole 删除角色。

相关信息

有关实例配置文件的基本信息，请参阅 使用实例配置文件。

有关服务相关角色的一般信息，请参阅创建服务相关角色。

担任角色的方法

您必须先对用户授予切换到您创建的角色的权限，然后用户、应用程序或服务才能使用该角色。您可使用附加到组或用户的任何策略授予所需权限。授予权限后，用户可以从 Amazon Web Services 管理控制台、适用于 Windows PowerShell 的工具、Amazon Command Line Interface（Amazon CLI）和 AssumeRole API 中代入角色。

> **Important**
>
> 当您以编程方式而不是在 IAM 控制台中创建角色，则除最长可达 64 个字符的 RoleName 外，您还可以选择添加最长 512 个字符的 Path。不过，如果您打算通过 Amazon Web Services 管理控制台 中的 Switch Role（切换角色）功能使用角色，则组合的 Path 和 RoleName 不能超过 64 个字符。

用于代入角色的方法决定了谁可以代入该角色，以及角色会话可以持续多长时间。使用 AssumeRole* API 操作时，您所担任的 IAM 角色为资源。调用 AssumeRole* API 操作的用户或角色是主体。

下表比较了代入角色的方法。

担任角色的方法	谁可以担任角色	用于指定凭证生命周期的方法	凭证生命周期 (最小值 | 最大值 | 默认值)
Amazon Web Services 管理控制台	用户或角色[1]（通过切换角色）	Role（角色）“Summary（摘要）”页面上的 Maximum session duration（最长会话持续时间）	15 分 | 最大会话持续时间设置[2] | 1 小时
assume-role CLI 或 AssumeRole API 操作	用户或角色[1]	duration-seconds CLI 或 DurationSeconds API 参数	15 分 | 最大会话持续时间设置[2] | 1 小时

担任角色的方法	谁可以担任角色	用于指定凭证生命周期的方法	凭证生命周期 (最小值 | 最大值 | 默认值)
assume-role-with-saml CLI 或 AssumeRoleWithSAML API 操作	使用 SAML 验证的任何用户	duration-seconds CLI 或 DurationSeconds API 参数	15 分 | 最大会话持续时间设置[2] | 1 小时
assume-role-with-web-identity CLI 或 AssumeRoleWithWebIdentity API 操作	使用 OIDC 提供者验证身份的任何用户	duration-seconds CLI 或 DurationSeconds API 参数	15 分 | 最大会话持续时间设置[2] | 1 小时
使用 构造的控制台 URLAssumeRole	用户或角色	URL 中的 SessionDuration HTML 参数	15 分钟 | 12 小时 | 1 小时
使用 构造的控制台 URLAssumeRoleWithSAML	使用 SAML 验证的任何用户	URL 中的 SessionDuration HTML 参数	15 分钟 | 12 小时 | 1 小时
使用 构造的控制台 URLAssumeRoleWithWebIdentity	使用 OIDC 提供者验证身份的任何用户	URL 中的 SessionDuration HTML 参数	15 分钟 | 12 小时 | 1 小时

[1] 使用一个角色的凭证代入其他角色称为角色链。使用角色链时，角色的会话持续时间限制为 1 小时。这适用于 Amazon Web Services 管理控制台 角色切换、Amazon CLI 和 API 操作。此限制不适用于最初通过用户凭证代入角色的人，也不适用于使用实例配置文件在 Amazon EC2 实例上运行的应用程序。

[2] 该设置可以具有 1 小时到 12 小时之间的值。有关修改最大会话持续时间设置的详细信息，请参阅 IAM 角色管理。该设置确定在获取角色凭证时可请求的最大会话持续时间。例如，在使用 AssumeRole* API 操作担任角色时，您可以使用 `DurationSeconds` 参数指定会话长度。可以使用该参数指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的角色会话长度。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或其用户会话中的剩余时间（以较少者为准）。假定您在角色上设置 5 小时的最大持续时间。已登录到控制台 10 小时（默认最多 12 小时）的 IAM 用户切换到该角色。可用角色会话持续时间为 2 小时。要了解如何查看您的角色的最大值，请参阅本页后面的更新角色的最长会话持续时间。

> **备注**
>
> - 最大会话持续时间设置不限制 Amazon 服务建立的会话。
> - Amazon EC2 IAM 角色凭证不受角色中配置的最长会话持续时间的限制。
> - 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源（例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda）的 Amazon Web Services 服务会提供临时凭证并自动更新这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 会话标签 或者 会话策略，则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN，请参阅 更新角色信任策略。

主题

- 从用户切换到 IAM 角色（控制台）
- 切换到 IAM 角色（Amazon CLI）
- 切换到 IAM 角色（Tools for Windows PowerShell）
- 切换到 IAM 角色（Amazon API）
- 使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限
- 使用实例配置文件

从用户切换到 IAM 角色（控制台）

以 IAM 用户、IAM Identity Center 中的用户、SAML 联合角色或 Web 联合身份角色登录时，您可以切换角色。角色 指定可用于访问所需的 Amazon 资源的一组权限。但是，您没有登录到角色，不过一旦以 IAM 用户身份登录，您就可以切换为 IAM 角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon Web Services 账户 中。有关角色、其权益以及如何创建角色的更多信息，请参阅IAM 角色和IAM 角色创建。

您的用户权限和切换为的角色的权限不会累积。一次只有一组权限处于活动状态。切换到一个角色后，您将临时放弃用户权限并使用分配给该角色的权限。退出该角色后，您的用户权限将自动恢复。

当您在 Amazon Web Services 管理控制台中切换角色时，控制台总是使用您的原始凭证对切换操作进行授权。例如，如果您切换到角色 A，则 IAM 使用您的原始凭证确定是否允许您担任角色 A。如果随后在使用角色 A 时尝试切换到角色 B，Amazon 仍会使用您的原始凭证对切换进行授权，而不是使用角色 A 的凭证。

> **Note**
>
> 当您以 IAM Identity Center 的用户、SAML 联合角色或 Web 身份联合角色登录时，您将在开始会话时代入 IAM 角色。例如，当 IAM Identity Center 中的用户登录 Amazon Web Services 访问门户时，他们必须选择与角色相关的权限集，然后才能访问 Amazon 资源。

角色会话

当您切换角色时，Amazon Web Services 管理控制台 会话默认持续 1 小时。IAM 用户会话默认为 12 小时，其他用户可能会定义不同的会话持续时间。在控制台内切换角色时，您将被授予角色最长会话持续时间或您的用户会话中的剩余时间（以较短者为准）。您不能通过代入角色来延长会话持续时间。

例如，假定角色的最长会话持续时间为 10 小时。当您决定切换到该角色时，您已登录控制台 8 小时。您的用户会话还剩 4 小时，因此允许的角色会话持续时间为 4 小时，而不是最长会话持续时间 10 小时。。下表显示了如何在控制台中切换角色时确定 IAM 用户的会话持续时间。

IAM 用户会话剩余时间...	角色会话持续时间...		
小于角色最长会话持续时间	用户会话中的剩余时间		

IAM 用户会话剩余时间...	角色会话持续时间...		
大于角色最长会话持续时间	等于最长会话持续时间值		
等于角色最长会话持续时间	等于最长会话持续时间值（近似值）		

使用一个角色的凭证代入其他角色称为角色链。使用角色链时，无论为各个角色配置的最长会话持续时间设置如何，会话持续时间都限制为 1 小时。这适用于 Amazon Web Services 管理控制台 角色切换、Amazon CLI 和 API 操作。

> **Note**
>
> 某些 Amazon 服务控制台可以在角色会话过期时自动续订角色会话，而无需您执行任何操作。有些可能会提示您重新加载浏览器页面以重新验证您的会话。

注意事项

- 如果您以 Amazon Web Services 账户根用户 身份登录，则无法切换角色。
- 必须向用户授予通过策略切换角色的权限。有关说明，请参阅向用户授予切换角色的权限。
- 您无法将 Amazon Web Services 管理控制台 中的角色切换到需要 ExternalId 值的角色。您只能通过调用支持 `ExternalId` 参数的 `AssumeRole` API 来切换到此类角色。

要切换为角色

1. 按照《Amazon 登录 用户指南》中的 Sign in to the Amazon Web Services 管理控制台 所述，根据用户类型选择相应的登录过程。
2. 在 Amazon Web Services 管理控制台 中，在右上角的导航栏上选择您的用户名。它通常类似于：*username*@*account_ID_number_or_alias*。
3. 选择以下方法之一来切换角色：
   - 选择切换角色。
   - 如果选择启用多会话支持，请选择添加会话，然后选择切换角色。

> **Note**
>
> 在 Amazon Web Services 管理控制台中的单个 Web 浏览器中，您最多可以同时登录五个不同的身份。有关详细信息，请参阅《Amazon Web Services 管理控制台入门指南》中的登录多个账户。

4. 在 Switch Role 页面上，键入账户 ID 号或账户别名以及管理员提供的角色的名称。

> **Note**
>
> 如果您的管理员创建了包含路径的角色（如 `division_abc/subdivision_efg/roleToDoX`），则必须在 Role 框中键入完整路径和名称。如果您仅键入角色名称，或组合的 `Path` 和 `RoleName` 超过 64 个字符，角色切换将失败。这是存储角色名称的浏览器 Cookie 的限制。如果发生这种情况，请与您的管理员联系，并要求他们减小路径和角色名称大小。

5. （可选）您可以输入显示名称，并选择在控制台导航栏中突出显示该角色的显示颜色。

   - 对于显示名称，键入您希望在此角色处于活动状态时将在导航栏上显示，以替代您的用户名的文本。系统会基于账户和角色信息提供建议名称，但是您可以将它更改为对您有意义的任何名称。
   - 在显示颜色中，选择用于突出显示名称的颜色。

   名称和颜色可帮助提醒您此角色处于活动状态的时间，这将更改您的权限。例如，对于向您提供对测试环境的访问权限的角色，您可以将 Display name（显示名称）指定为 **`Test`**，并选择绿色的 Color（颜色）。对于向您授予对生产环境的访问权限的角色，您可以将 Display name（显示名称）指定为 **`Production`**，并选择红色作为 Color（颜色）。

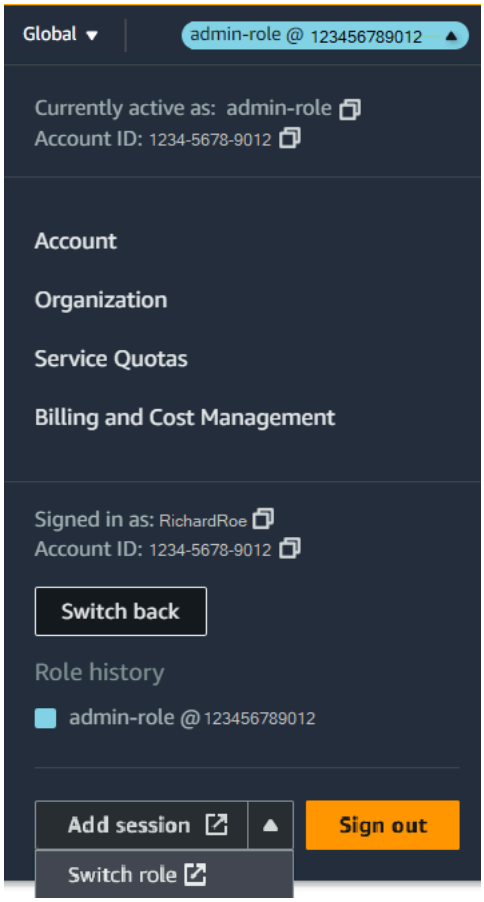
6. 选择切换角色。显示名称和颜色会在导航栏上替换您的用户名，您可以开始使用角色向您授予的权限。

7. 完成需要 IAM 角色的任务之后，您可以切换回原始会话。这将移除该角色提供的其他权限，并使您恢复到标准权限。

   a. 在 IAM 控制台中，在右上角的导航栏中选择角色的 Display Name（显示名称）。

   b. 选择切换回。

例如，假设您使用用户名 123456789012 登录账号 Richard。在使用 admin-role 角色后，您要停止使用该角色并返回到您的原始权限。要停止使用该角色，请选择 admin-role @ 123456789012，然后选择切换回。



> **Tip**
>
> 您使用的最后几个角色将显示在菜单上。下次想要切换到其中的一个角色时，您可以直接选择所需的角色。只有在菜单上未显示角色时，您才需要手动键入账户和角色信息。

其他资源

- 向用户授予切换角色的权限
- 向用户授予权限以将角色传递给 Amazon 服务
- 创建角色，向 IAM 用户授予权限
- 创建向 Amazon 服务委派权限的角色

- 排查 IAM 角色问题

切换到 IAM 角色（Amazon CLI）

角色 指定可用于访问所需的 Amazon 资源的一组权限。在这种意义上，它类似于 Amazon Identity and Access Management 中的用户 (IAM)。作为用户登录时，您会获取一组特定权限。不过，您不会登录到角色，但在以用户身份登录后，您可以切换到角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon Web Services 账户 中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅IAM 角色和IAM 角色创建。要了解在担任角色时使用的各种方法，请参阅担任角色的方法。

> **Important**
>
> 不会累积您的 IAM 用户和担任的任何角色的权限。一次只有一组权限处于活动状态。在担任某个角色时，您将临时放弃以前的用户或角色权限并使用为该角色分配的权限。退出该角色后，您的用户权限将自动恢复。

在以 IAM 用户身份登录后，您可以使用角色来运行 Amazon CLI 命令。在以已使用角色的 externally authenticated user（外部验证的用户）（SAML 或 OIDC）身份登录后，您也可以使用角色来运行 Amazon CLI 命令。此外，您还可以使用角色从通过实例配置文件附加到角色的 Amazon EC2 实例中运行 Amazon CLI 命令。在以 Amazon Web Services 账户根用户身份登录时，您无法担任角色。

角色链 — 您还可以使用角色链，这将使用来自一个角色的权限访问另一个角色。

默认情况下，您的角色会话持续 1 小时。在使用 `assume-role*` CLI 操作担任该角色时，您可以为 `duration-seconds` 参数指定一个值。该值的范围在 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间。如果您在控制台中切换角色，则会话持续时间最长为一小时。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。

如果使用角色链，您的会话持续时间限制为最多 1 小时。如果您随后使用 `duration-seconds` 参数提供大于 1 小时的值，操作将失败。

示例方案：切换到生产角色

假设您是一名在开发环境中工作的 IAM 用户。在此场景中，您有时需要使用 Amazon CLI 通过命令行来使用生产环境。您已经有一组可用的访问密钥凭证。这可能是分配给您的标准 IAM 用户的访问密钥对。或者，如果您以 SAML 或 OIDC 联合主体身份登录，则它可能是最初为您分配的角色的访问

密钥对。如果您的当前权限授予您担任特定 IAM 角色的能力，则可以在 Amazon CLI 配置文件的“配置文件”中标识该角色。然后，将使用指定 IAM 角色而非原始身份的权限运行该命令。请注意，通过 Amazon CLI 命令指定该配置文件时，您使用的是新角色。在这种情况下，您无法同时使用开发账户中的原始权限。原因是，一次仅一组权限能够生效。

> **Note**
>
> 为了安全起见，管理员可以查看 Amazon CloudTrail 日志以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅sts:SourceIdentity和sts:RoleSessionName。

切换到生产角色 (Amazon CLI)

1. 如果您从未使用过 Amazon CLI，则您必须先配置默认的 CLI 配置文件。打开命令提示符并将您的 Amazon CLI 安装设置为使用来自 IAM 用户或联合角色的访问密钥。有关更多信息，请参阅《Amazon Command Line Interface 用户指南》中的配置 Amazon Command Line Interface。

   运行 aws configure 命令，如下所述：

   ```
   aws configure
   ```

   当系统提示时，请提供以下信息：

   ```
   AWS Access Key ID [None]: AKIAIOSFODNN7EXAMPLE
   AWS Secret Access Key [None]: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
   Default region name [None]: us-east-2
   Default output format [None]: json
   ```

2. 在 Unix 或 Linux 上的 `.aws/config` 文件或 Windows 上的 `C:\Users\USERNAME\.aws\config` 文件中，为角色创建新的配置文件。以下示例创建一个名为 `prodaccess` 的配置文件，它切换到 123456789012 账户中的 *ProductionAccessRole* 角色。您从创建该角色的账户管理员处获取角色 ARN。在调用此配置文件时，Amazon CLI 使用 `source_profile` 的凭证请求该角色的凭证。因此，引用为 `source_profile` 的身份必须具有 `role_arn` 中指定的角色的 `sts:AssumeRole` 权限。

   ```
   [profile prodaccess]
       role_arn = arn:aws:iam::123456789012:role/ProductionAccessRole
       source_profile = default
   ```

3. 在创建新的配置文件后，将使用用附加到 IAM 角色 ProductionAccessRole（而不是默认用户）的权限运行指定 --profile prodaccess 参数的任何 Amazon CLI 命令。

```
aws iam list-users --profile prodaccess
```

   如果分配给 ProductionAccessRole 的权限允许列出当前 Amazon 账户中的用户，则此命令有效。

4. 要返回到原始凭证授予的权限，请运行不带 --profile 参数的命令。Amazon CLI 将恢复使用您在Step 1中配置的默认配置文件中的凭证。

有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的代入角色。

示例场景：允许实例配置文件角色切换到另一个账户中的角色

假设您使用两个 Amazon Web Services 账户，并希望允许 Amazon EC2 实例上运行的应用程序在两个账户中运行 Amazon CLI 命令。假设 EC2 实例位于 111111111111 账户中。该实例包含 abcd 实例配置文件角色，以允许应用程序对同一 111111111111 账户中的 amzn-s3-demo-bucket1 存储桶执行只读 Amazon S3 任务。不过，还必须允许应用程序担任 efgh 跨账户角色以在账户 222222222222 中执行任务。为此，abcd EC2 实例配置文件角色必须具有以下权限策略：

账户 111111111111 ***abcd*** 角色权限策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
        }
    ]
}
```

假设 efgh 跨账户角色允许对同一 222222222222 账户中的 amzn-s3-demo-bucket2 存储桶执行只读 Amazon S3 任务。为此，efgh 跨账户角色必须具有以下权限策略：

账户 222222222222 ***efgh*** 角色权限策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
```

```
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket2/*",
                "arn:aws:s3:::amzn-s3-demo-bucket2"
            ]
        }
    ]
}
```

efgh 角色必须允许 abcd 实例配置文件角色担任该角色。为此，efgh 角色必须具有以下信任策略：

账户 222222222222 ***efgh*** 角色信任策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

要在 222222222222 账户中运行 Amazon CLI 命令，您必须更新 CLI 配置文件。在 Amazon CLI 配置文件中将 efgh 角色指定为“配置文件”，并将 abcd EC2 实例配置文件角色指定为“凭证源”。然后，将使用 efgh 角色的权限运行 CLI 命令，而不是使用原始 abcd 角色。

> **Note**
>
> 出于安全原因，您可以使用 Amazon CloudTrail 审核角色在账户中的使用。要在 CloudTrail 日志中由不同主体使用角色时区分角色会话，可以借助角色会话名称。在 Amazon CLI 按照

本主题所述代表用户担任某个角色时，将自动创建一个 Amazon-CLI-session-*nnnnnnnn* 形式的角色会话名称。其中，*nnnnnnnn* 是一个表示 Unix 纪元时间 (自 1970 年 1 月 1 日午夜 UTC 算起的秒数) 的整数。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的 CloudTrail 事件引用。

允许 EC2 实例配置文件角色切换到跨账户角色 (Amazon CLI)

1. 您不必配置默认 CLI 配置文件。相反，您可以从 EC2 实例配置文件元数据中加载凭证。在 `.aws/config` 文件中为角色创建新的配置文件。以下示例创建一个 `instancecrossaccount` 配置文件，它切换到 222222222222 账户中的 *efgh* 角色。在调用该配置文件时，Amazon CLI 使用 EC2 实例配置文件元数据的凭证请求该角色的凭证。因此，EC2 实例配置文件角色必须具有 `role_arn` 中指定的角色的 `sts:AssumeRole` 权限。

   ```
   [profile instancecrossaccount]
   role_arn = arn:aws:iam::222222222222:role/efgh
   credential_source = Ec2InstanceMetadata
   ```

2. 在创建新的配置文件后，将使用用附加到 222222222222 账户中的 efgh 角色的权限运行指定 `--profile instancecrossaccount` 参数的任何 Amazon CLI 命令。

   ```
   aws s3 ls amzn-s3-demo-bucket2 --profile instancecrossaccount
   ```

   如果分配给 efgh 角色的权限允许列出当前 Amazon Web Services 账户 中的用户，则该命令有效。

3. 要恢复为 111111111111 账户中的原始 EC2 实例配置文件权限，请不要使用 `--profile` 参数运行 CLI 命令。

有关更多信息，请参阅 Amazon Command Line Interface 用户指南中的代入角色。

切换到 IAM 角色（Tools for Windows PowerShell）

角色 指定可用于访问所需的 Amazon 资源的一组权限。在这种意义上，它类似于 Amazon Identity and Access Management 中的用户 (IAM)。作为用户登录时，您会获取一组特定权限。但是，您没有登录到角色，不过一旦登录，您就可以切换为角色。这会临时搁置原始用户权限，而向您提供分配给角色的权限。角色可以在您自己的账户中或任何其他 Amazon Web Services 账户 中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅IAM 角色和IAM 角色创建。

> **Important**
>
> 您的 IAM 用户权限和切换为的角色的权限不会累积。一次只有一组权限处于活动状态。切换到一个角色后，您将临时放弃用户权限并使用分配给该角色的权限。退出该角色后，您的用户权限将自动恢复。

本部分介绍如何在通过 Amazon Tools for Windows PowerShell 在命令行上工作时切换角色。

假设您在开发环境中有一个账户，并且您有时需要使用 Tools for Windows PowerShell 在命令行上操作生产环境。您已经有一组可用的访问密钥凭证。这些凭证可以是分配给您的标准 IAM 用户的访问密钥对。或者，如果您以 SAML 或 OIDC 联合主体身份登录，则其可以是最初分配给您的角色的访问密钥对。您可以使用这些凭证运行 `Use-STSRole` cmdlet，这会将新角色的 ARN 作为参数传递。该命令将返回请求的角色的临时安全凭证。然后可利用该角色的权限在后续 PowerShell 命令中使用那些凭证来使用访问生产中的资源。在使用角色时，您不能使用开发账户中的用户权限，因为一次只有一组权限有效。

> **Note**
>
> 为了安全起见，管理员可以查看 Amazon CloudTrail 日志以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅`sts:SourceIdentity`和`sts:RoleSessionName`。

请注意，所有访问密钥和令牌都只是示例，不能原样照用。请用您的实际环境的适当值替换。

要切换到某个角色 (Tools for Windows PowerShell)

1. 打开 PowerShell 命令提示符并将您的默认配置文件配置为使用来自您的当前 IAM 用户或联合角色的访问密钥。如果您以前使用过 Tools for Windows PowerShell，则可能已完成此设置。请注意，您只能在以 IAM 用户身份（而非 Amazon Web Services 账户根用户 身份）登录时切换角色。

   ```
   PS C:\> Set-AWSCredentials -AccessKey AKIAIOSFODNN7EXAMPLE -
   SecretKey wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY -StoreAs MyMainUserProfile
   PS C:\> Initialize-AWSDefaults -ProfileName MyMainUserProfile -Region us-east-2
   ```

   有关更多信息，请参阅 Amazon Tools for PowerShell 用户指南中的使用 Amazon 凭证。

2. 要检索新角色的凭证，请运行以下命令切换到 123456789012 账户中的 *RoleName* 角色。您从创建该角色的账户管理员处获取角色 ARN。该命令还需要您提供会话名称。您可以选择该名称的任何文本。以下命令请求凭证，然后从返回的结果对象中捕获 Credentials 属性对象并将其存储在 $Creds 变量中。

```
PS C:\> $Creds = (Use-STSRole -RoleArn "arn:aws:iam::123456789012:role/RoleName" -
RoleSessionName "MyRoleSessionName").Credentials
```

$Creds 是一个对象，现在包含您在后续步骤中所需的 AccessKeyId、SecretAccessKey 和 SessionToken 元素。以下示例命令说明典型值：

```
PS C:\> $Creds.AccessKeyId
AKIAIOSFODNN7EXAMPLE

PS C:\> $Creds.SecretAccessKey
wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY

PS C:\> $Creds.SessionToken
AQoDYXdzEGcaEXAMPLE2gsYULo
+Im5ZEXAMPLEeYjs1M2FUIgIJx9tQqNMBEXAMPLECvSRyh0FW7jEXAMPLEW+vE/7s1HRp
XviG7b+qYf4nD00EXAMPLEmj4wxS04L/uZEXAMPLECihzFB5lTYLto9dyBgSDyEXAMPLE9/
g7QRUhZp4bqbEXAMPLENwGPy
Oj59pFA4lNKCIkVgkREXAMPLEjlzxQ7y52gekeVEXAMPLEDiB9ST3UuysgsKdEXAMPLE1TVastU1A0SKFEXAMPLEiyw
C
s8EXAMPLEpZgOs+6hz4AP4KEXAMPLERbASP+4eZScEXAMPLEsnf87eNhyDHq6ikBQ==

PS C:\> $Creds.Expiration
Thursday, June 18, 2018 2:28:31 PM
```

3. 要在任何后续命令中使用这些凭证，应使用 -Credential 参数来包含它们。例如，以下命令使用来自角色的凭证，并只有在角色被授予 iam:ListRoles 权限时，命令才会起作用并可以因此运行 Get-IAMRoles cmdlet：

```
PS C:\> get-iamroles -Credential $Creds
```

4. 要返回到您的原始凭证，只需停止使用 -Credentials $Creds 参数并允许 PowerShell 恢复为在默认配置文件中存储的凭证。

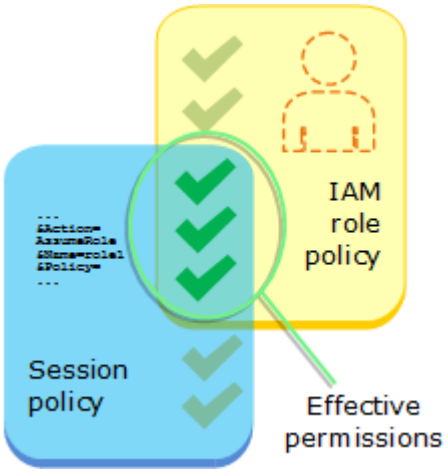
切换到 IAM 角色（Amazon API）

角色 指定可用于访问 Amazon 资源的一组权限。在这种意义上，它类似于 IAM 用户。主体（用户或应用程序）将担任角色以获得临时权限，从而执行所需任务并与 Amazon 资源交互。角色可以在您自己的账户中或任何其他 Amazon Web Services 账户 中。有关角色、其权益以及如何创建和配置角色的更多信息，请参阅IAM 角色和IAM 角色创建。要了解在担任角色时使用的各种方法，请参阅担任角色的方法。

> **Important**
>
> 不会累积您的 IAM 用户和担任的任何角色的权限。一次只有一组权限处于活动状态。在担任某个角色时，您将临时放弃以前的用户或角色权限并使用为该角色分配的权限。退出该角色后，您的原始权限将自动恢复。

要担任角色，应用程序需调用 Amazon STS `AssumeRole` API 操作并传递角色的 ARN 以供使用。该操作将使用临时凭证创建一个新会话。此会话与用于该角色的基于身份的策略具有相同的权限。

在调用 `AssumeRole` 时，您可以选择传递内联或托管会话策略。会话策略是高级策略，在以编程方式为角色或联合用户会话创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 `Policy` 参数传递单个 JSON 内联会话策略文档。您可以使用 `PolicyArns` 参数指定最多 10 个托管会话策略。生成的会话的权限是实体的基于身份的策略与会话策略的交集。如果需要为其他人提供角色的临时凭证，会话策略是非常有用的。他们可以在后续的 Amazon API 调用中使用角色的临时凭证来访问拥有该角色的账户中的资源。您使用会话策略授予的权限不能超过基于身份的策略允许的权限。要了解有关 Amazon 如何确定角色的有效权限的更多信息，请参阅策略评估逻辑。



在以 IAM 用户或已使用角色的 externally authenticated user（外部验证的用户（SAML 或 OIDC）身份登录后，您可以调用 AssumeRole。您还可以使用角色链，它使用一个角色担任另一个角色。在以 Amazon Web Services 账户根用户身份登录时，您无法担任角色。

默认情况下，您的角色会话持续 1 小时。在使用 Amazon STS AssumeRole* API 操作担任该角色时，您可以为 DurationSeconds 参数指定一个值。该值的范围在 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。

如果使用角色链，您的会话限制为最多 1 小时。如果您随后使用 DurationSeconds 参数提供大于 1 小时的值，操作将失败。

> **Note**
>
> 为了安全起见，管理员可以查看 Amazon CloudTrail 日志以了解已在 Amazon 中执行操作的人员。您的管理员可能会要求您在代入角色时指定源身份或角色会话名称。有关更多信息，请参阅sts:SourceIdentity和sts:RoleSessionName。

以下代码示例演示了如何创建用户并代入角色。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 S3 存储桶，然后清除资源。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
```

```
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
```

```
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
```

```
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
 DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
```

```
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
```

```
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
 DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
```

```
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }
```

```
        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
```

```
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
 ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
```

```
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
 ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
 ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };
```

```
        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
accessKeyId });
```

```
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;

namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();
```

```
        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();

        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
```

```
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");
```

```
        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);
```

```
        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
complete.");
        Console.WriteLine("Please wait while we clean up the resources we
created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
patience.");
    }
```

```
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
```

```
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
{ BucketName = bucketName });
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
```

```
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
 { BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
```

```
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }
```

```
    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
```

```
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
###############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
```

```
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
```

```
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
  fi

  local assume_role_policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

  local assume_role_policy_name
  assume_role_policy_name=$(generate_random_name "test-assume-role-")

  # shellcheck disable=SC2181
  local assume_role_policy_arn
  assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
  else
    errecho "The policy failed to create."
```

```
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
    return 1
  fi

  echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
  sleep 10
  echo_repeat "*" 88
  echo

  echo "Try to list buckets without the new user assuming the role."
  echo_repeat "*" 88
  echo

  # Set the environment variables for the created user.
  # bashsupport disable=BP2001
  export AWS_ACCESS_KEY_ID=$key_name
  # bashsupport disable=BP2001
  export AWS_SECRET_ACCESS_KEY=$key_secret

  local buckets
  buckets=$(s3_list_buckets)

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. This should not have
happened."
  else
    errecho "Because the role with permissions has not been assumed, listing
buckets failed."
  fi

  echo
  echo_repeat "*" 88
  echo "Now assume the role $iam_role_name and list the buckets."
  echo_repeat "*" 88
  echo

  local credentials

  credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
```

```
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. Listing buckets
succeeded because of "
   echo "the assumed role."
 else
   errecho "Failed to list buckets. This should not happen."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   export AWS_SESSION_TOKEN=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 local result=0
 export AWS_ACCESS_KEY_ID=""
 export AWS_SECRET_ACCESS_KEY=""

 echo
 echo_repeat "*" 88
```

```
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

此场景中使用的 IAM 函数。

```
########################################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
########################################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}
```

```
  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
```

```
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    User name:   $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
  errecho "ERROR: A user with that name already exists in the account."
  return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
  --output text \
  --query 'User.Arn')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-user operation failed.$response"
  return 1
fi

echo "$response"
```

```
  return 0
}

################################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
################################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
```

```
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
```

```
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-role \
    --role-name "$role_name" \
    --assume-role-policy-document "$policy_document" \
    --output text \
    --query Role.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_create_policy"
  echo "Creates an AWS Identity and Access Management (IAM) policy."
  echo "  -n policy_name   The name of the IAM policy."
  echo "  -p policy_json -- The policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) policy_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_name" ]]; then
  errecho "ERROR: You must provide a policy name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-policy \
  --policy-name "$policy_name" \
  --policy-document "$policy_document" \
  --output text \
```

```
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
```

```
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

########################################################################################
```

```
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1
```

```
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.
```

```
# bashsupport disable=BP5008
function usage() {
  echo "function iam_delete_policy"
  echo "Deletes an AWS Identity and Access Management (IAM) policy"
  echo "  -n policy_arn -- The name of the IAM policy arn."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Policy arn:  $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
```

```
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
```

```
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
```

```
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      k) access_key="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  if [[ -z "$access_key" ]]; then
    errecho "ERROR: You must provide an access key with the -k parameter."
    usage
```

```
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Username:   $user_name"
  iecho "    Access key:   $access_key"
  iecho ""

  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
 }

 # Retrieve the calling parameters.
 while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
 done
 export OPTIND=1

 if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
 fi

 iecho "Parameters:\n"
 iecho "    User name:   $user_name"
 iecho ""

 # If the user does not exist, we don't want to try to delete it.
 if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
 fi

 response=$(aws iam delete-user \
    --user-name "$user_name")

 local error_code=${?}
```

```
  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }

    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
```

```
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
std::endl;
        }

        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;
```

```
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
```

```
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }
```

```
        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);

        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
```

```
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());
```

```
        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }

    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
```

```
            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
    bool result = true;
    if (policy.ArnHasBeenSet()) {
        // Detach the policy from the role.
        {
            Aws::IAM::Model::DetachRolePolicyRequest request;
            request.SetPolicyArn(policy.GetArn());
            request.SetRoleName(role.GetRoleName());

            Aws::IAM::Model::DetachRolePolicyOutcome outcome =
 client.DetachRolePolicy(
                    request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error Detaching policy from roles. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully detached the policy with arn "
                          << policy.GetArn()
                          << " from role " << role.GetRoleName() << "." <<
 std::endl;
            }
        }
```

```
        // Delete the policy.
        {
            Aws::IAM::Model::DeletePolicyRequest request;
            request.WithPolicyArn(policy.GetArn());

            Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error deleting policy. " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = false;
            }
            else {
                std::cout << "Successfully deleted the policy with arn "
                          << policy.GetArn() << std::endl;
            }
        }

    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
```

```
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
```

```
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser(ctx)
 accessKey := scenario.CreateAccessKey(ctx, user)
 role := scenario.CreateRoleAndPolicies(ctx, user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
 scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
 scenario.Cleanup(ctx, user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}

// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(ctx, userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
```

```
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context,
 user *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"},
 "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
```

```
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
  ))
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

 s3Client := s3.NewFromConfig(noPermsConfig)
 _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
```

```
      "the ListBuckets call was made without permissions.")
    default:
     log.Println("Expected AccessDenied, got something else.")
     panic(err)
    }
   }
  } else {
   log.Println("Expected AccessDenied error when calling ListBuckets without
  permissions,\n" +
    "but the call succeeded. Continuing the example anyway...")
  }
  log.Println(strings.Repeat("-", 88))
  return &noPermsConfig
}

// ListBucketsWithAssumedRole performs the following actions:
//
//  1. Creates an AWS Security Token Service (AWS STS) client from the config
  created from
//     the user's access key credentials.
//  2. Gets temporary credentials by assuming the role that grants permission to
  list the
//     buckets.
//  3. Creates an Amazon S3 client from the temporary credentials.
//  4. Lists buckets for the account. Because the temporary credentials are
  generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx
  context.Context, noPermsConfig *aws.Config, role *types.Role) {
  log.Println("Let's assume the role that grants permission to list buckets and
  try again.")
  scenario.questioner.Ask("Press Enter when you're ready.")
  stsClient := sts.NewFromConfig(*noPermsConfig)
  tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
   RoleArn:         role.Arn,
   RoleSessionName: aws.String("AssumeRoleExampleSession"),
   DurationSeconds: aws.Int32(900),
  })
  if err != nil {
   log.Printf("Couldn't assume role %v.\n", *role.RoleName)
   panic(err)
  }
  log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
  assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
```

```
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)

 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
   err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
 *policy.PolicyArn)
   if err != nil {
```

```
    panic(err)
   }
   err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   log.Printf("Detached policy %v from role %v and deleted the policy.\n",
    *policy.PolicyName, *role.RoleName)
  }
  err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted role %v.\n", *role.RoleName)

  userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, userPol := range userPols {
   err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
  }
  keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, key := range keys {
   err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName,
*key.AccessKeyId)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
*user.UserName)
  }
  err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
```

```
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
 Pause(secs int)
}

const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

定义一个封装账户操作的结构。

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

定义一个封装策略操作的结构。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
```

```
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
```

```
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

定义一个封装角色操作的结构。

```
import (
 "context"
 "encoding/json"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
```

```
    Principal: map[string]string{"AWS": trustedUserArn},
    Action:    []string{"sts:AssumeRole"},
   }},
  }
  policyBytes, err := json.Marshal(trustPolicy)
  if err != nil {
   log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
  trustedUserArn, err)
   return nil, err
  }
  result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
   AssumeRolePolicyDocument: aws.String(string(policyBytes)),
   RoleName:                 aws.String(roleName),
  })
  if err != nil {
   log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
  } else {
   role = result.Role
  }
  return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
```

```
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
```

```
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
```

```
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

定义一个封装用户操作的结构。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
```

```
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
```

```
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
```

```
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
```

```
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole

- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 用户操作的函数。

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
```

```
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static String userArn;

    public static void main(String[] args) throws Exception {

        final String usage = """

                Usage:
                    <username> <policyName> <roleName> <roleSessionName>
 <bucketName>\s

                Where:
                    username - The name of the IAM user to create.\s
                    policyName - The name of the policy to create.\s
                    roleName - The name of the role to create.\s
                    roleSessionName - The name of the session required for the
 assumeRole operation.\s
                    bucketName - The name of the Amazon S3 bucket from which
 objects are read.\s
                """;

        if (args.length != 5) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
```

```
        String policyName = args[1];
        String roleName = args[2];
        String roleSessionName = args[3];
        String bucketName = args[4];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the AWS IAM example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
```

```
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
```

```
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();
```

```
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();

            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
```

```
   public static void deleteRole(IamClient iam, String roleName, String polArn)
{

       try {
           // First the policy needs to be detached.
           DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                   .policyArn(polArn)
                   .roleName(roleName)
                   .build();

           iam.detachRolePolicy(rolePolicyRequest);

           // Delete the policy.
           DeletePolicyRequest request = DeletePolicyRequest.builder()
                   .policyArn(polArn)
                   .build();

           iam.deletePolicy(request);
           System.out.println("*** Successfully deleted " + polArn);

           // Delete the role.
           DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           iam.deleteRole(roleRequest);
           System.out.println("*** Successfully deleted " + roleName);

       } catch (IamException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static void deleteKey(IamClient iam, String username, String
accessKey) {
       try {
           DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                   .accessKeyId(accessKey)
                   .userName(username)
                   .build();

           iam.deleteAccessKey(request);
```

```
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
```

```
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
```

```
 }

 // Create an access key. This key is used to authenticate the new user to
 // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
(AWS STS).
 // It's not best practice to use access keys. For more information, see
https://aws.amazon.com/iam/resources/best-practices/.
 const createAccessKeyResponse = await iamClient.send(
   new CreateAccessKeyCommand({ UserName: userName }),
 );

 if (
   !createAccessKeyResponse.AccessKey?.AccessKeyId ||
   !createAccessKeyResponse.AccessKey?.SecretAccessKey
 ) {
   throw new Error("Access key not created");
 }

 const {
   AccessKey: { AccessKeyId, SecretAccessKey },
 } = createAccessKeyResponse;

 let s3Client = new S3Client({
   credentials: {
     accessKeyId: AccessKeyId,
     secretAccessKey: SecretAccessKey,
   },
 });

 // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
 // thrown while the user and access keys are still stabilizing.
 await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
   try {
     return await listBuckets(s3Client);
   } catch (err) {
     if (err instanceof Error && err.name === "InvalidAccessKeyId") {
       throw err;
     }
   }
 });

 // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
 // is thrown while the user and access keys are still stabilizing.
```

```
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
                  // Allow the previously created user to assume this role.
                  AWS: User.Arn,
                },
                Action: "sts:AssumeRole",
              },
            ],
          }),
          RoleName: roleName,
        }),
      ),
  );

  if (!Role) {
    throw new Error("Role not created");
  }

  // Create a policy that allows the user to list S3 buckets.
  const { Policy: listBucketPolicy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Action: ["s3:ListAllMyBuckets"],
            Resource: "*",
          },
        ],
      }),
      PolicyName: policyName,
```

```
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);

// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
```

```
      accessKeyId: Credentials.AccessKeyId,
      secretAccessKey: Credentials.SecretAccessKey,
      sessionToken: Credentials.SessionToken,
    },
  });

  // List the S3 buckets again.
  // Retry the list buckets operation until it succeeds. AccessDenied might
  // be thrown while the role policy is still stabilizing.
  await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
    listBuckets(s3Client),
  );

  // Clean up.
  await iamClient.send(
    new DetachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
```

```
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }

  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

创建包装 IAM 用户操作的函数。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
 <bucketName>

    Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
 operation.
        fileLocation - The file location to the JSON required to create the role
 (see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
 read.
    """

    if (args.size != 6) {
        println(usage)
        exitProcess(1)
    }

    val userName = args[0]
    val policyName = args[1]
    val roleName = args[2]
    val roleSessionName = args[3]
    val fileLocation = args[4]
    val bucketName = args[5]

    createUser(userName)
```

```
    println("$userName was successfully created.")

    val polArn = createPolicy(policyName)
    println("The policy $polArn was successfully created.")

    val roleArn = createRole(roleName, fileLocation)
    println("$roleArn was successfully created.")
    attachRolePolicy(roleName, polArn)

    println("*** Wait for 1 MIN so the resource is available.")
    delay(60000)
    assumeGivenRole(roleArn, roleSessionName, bucketName)

    println("*** Getting ready to delete the AWS resources.")
    deleteRole(roleName, polArn)
    deleteUser(userName)
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
```

```
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
```

```
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
```

```
    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }

    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
```

```
    return jsonParser.parse(reader)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
```

```
use Iam\IAMService;

echo("\n");
echo("--------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("--------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
```

```
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}

$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
```

```
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
import json
import sys
import time
```

```
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, roles, and
 policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
```

```
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
```

```
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
"
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
"
            f"{error.response['Error']['Message']}"
        )
```

```
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
```

```
                              grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
```

```
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
```

```
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
```

```
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end

  # Creates a role that can be assumed by a user.
  #
  # @param role_name [String] The name to give the role.
  # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
  # @return [Aws::IAM::Role] The newly created role.
  def create_role(role_name, user)
    trust_policy = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Principal: { 'AWS': user.arn },
        Action: 'sts:AssumeRole'
      }]
    }.to_json
    role = @iam_client.create_role(
      role_name: role_name,
      assume_role_policy_document: trust_policy
    ).role
    @logger.info("Created role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a role for the demo. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
```

```
    raise
  else
    role
  end

  # Creates a policy that grants permission to list S3 buckets in the account,
and
  # then attaches the policy to a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param role [Aws::IAM::Role] The role that the policy is attached to.
  # @return [Aws::IAM::Policy] The newly created policy.
  def create_and_attach_role_policy(policy_name, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 's3:ListAllMyBuckets',
        Resource: 'arn:aws:s3:::*'
      }]
    }.to_json
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
```

```
 def create_user_policy(policy_name, user, role)
   policy_document = {
     Version: '2012-10-17',
     Statement: [{
       Effect: 'Allow',
       Action: 'sts:AssumeRole',
       Resource: role.arn
     }]
   }.to_json
   @iam_client.put_user_policy(
     user_name: user.user_name,
     policy_name: policy_name,
     policy_document: policy_document
   )
   puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
```

```
      break if count.zero?
    end
  rescue Aws::Errors::ServiceError => e
    if e.code == 'AccessDenied'
      puts('Attempt to list buckets with no permissions: AccessDenied.')
    else
      @logger.info("Couldn't list buckets for the account. Here's why: ")
      @logger.info("\t#{e.code}: #{e.message}")
      raise
    end
  end

  # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
  # This is separated into a factory function so that it can be mocked for unit
testing.
  #
  # @param key_id [String] The ID of the access key used by the STS client.
  # @param key_secret [String] The secret part of the access key used by the STS
client.
  def create_sts_client(key_id, key_secret)
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
```

```
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
 role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
 policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
 why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
 deleted
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
```

```
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;
```

```
    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
```

```
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;

    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );
```

```
   let attach_role_policy_result =
       iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
           .await?;
   println!(
       "Attached the policy to the role: {:?}",
       attach_role_policy_result
   );

   let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
   let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
   iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
       .await?;
   println!("Created inline policy.");

   //First, fail to list the buckets with the user.
   let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
   let fail_config = aws_config::from_env()
       .credentials_provider(creds.clone())
       .load()
       .await;
   println!("Fail config: {:?}", fail_config);
   let fail_client: s3Client = s3Client::new(&fail_config);
   match fail_client.list_buckets().send().await {
       Ok(e) => {
           println!("This should not run. {:?}", e);
       }
       Err(e) => {
           println!("Successfully failed with error: {:?}", e)
       }
   }

   let sts_config = aws_config::from_env()
       .credentials_provider(creds.clone())
       .load()
       .await;
   let sts_client: stsClient = stsClient::new(&sts_config);
   sleep(Duration::from_secs(10)).await;
   let assumed_role = sts_client
       .assume_role()
```

```
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
```

```
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy

- DetachRolePolicy
- PutUserPolicy

使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限

在 Amazon EC2 实例上运行的应用程序必须将 Amazon 凭证包含在其 Amazon API 请求中。您可以让开发人员将 Amazon 凭证直接存储在 Amazon EC2 实例中，并允许该实例中的应用程序使用这些凭证。但之后开发人员必须管理凭证，确保能安全地将凭证传递给每个实例，并在需要更新凭证时更新每个 Amazon EC2 实例。这需要进行大量的额外工作。

相反，您可以且应使用 IAM 角色管理在 Amazon EC2 实例上运行的应用程序的临时凭证。在使用角色时，您不需要将长期凭证（如登录凭证或访问密钥）分配给 Amazon EC2 实例。相反，角色可提供临时权限供应用程序在调用其他 Amazon 资源时使用。当您启动 Amazon EC2 实例时，可指定要与实例关联的 IAM 角色。然后，实例上运行的应用程序可使用角色提供的临时凭证对 API 请求进行签名。

若要使用角色向 Amazon EC2 实例上运行的应用程序授予权限，需要进行一点额外配置。Amazon EC2 实例上运行的应用程序由虚拟化操作系统从 Amazon 中提取。因为存在这一额外分离操作，所以需要执行一个附加步骤将 Amazon 角色及其关联权限分配给 Amazon EC2 实例，并使这些权限对其应用程序可用。此额外步骤是创建要附加到实例的实例配置文件。实例配置文件包含角色，并且可以为实例上运行的应用程序提供角色的临时凭证。然后，可以在应用程序的 API 调用中使用这些临时凭证访问资源，以及将访问限制为仅角色指定的那些资源。

> **Note**
>
> 一次只能将一个角色分配给一个 Amazon EC2 实例，实例上的所有应用程序都具有相同的角色和权限。使用 Amazon ECS 来管理 Amazon EC2 实例时，您可以向 Amazon ECS 任务分配角色，这些角色可以与其运行的 Amazon EC2 实例的角色不同。为每项任务分配角色符合最低权限访问的原则，并有利于对操作和资源进行更精细的控制。
> 有关更多信息，请参阅《Amazon Elastic Container Service 最佳实践指南》中的 Using IAM roles with Amazon ECS tasks。

以这种方式使用该角色拥有多种优势。因为角色凭证是临时性的，并且会自动更新，所以无需管理凭证，也不必担心长期安全风险。此外，如果您对多个实例使用单个角色，则可以对该角色进行更改，而此更改会自动传播到所有实例。

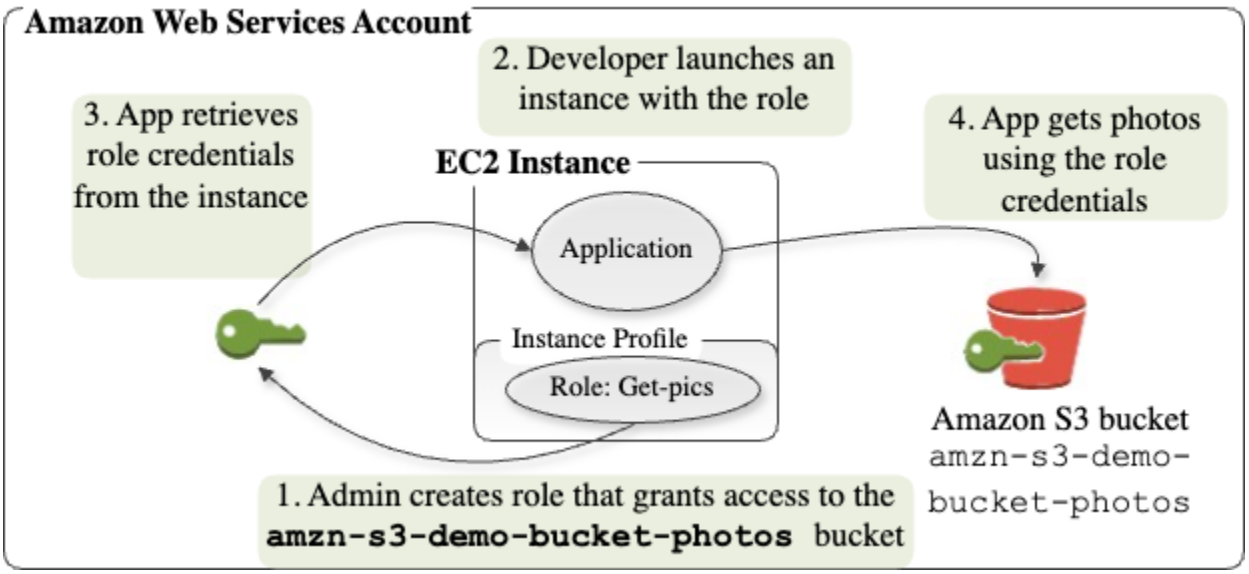
> **Note**
>
> 通常会在您启动 Amazon EC2 实例时向它分配角色，不过也可以向已在运行的 Amazon EC2 实例附加角色。要了解如何将角色附加到运行的实例，请参阅适用于 Amazon EC2 的 IAM 角色。

主题

- Amazon EC2 实例的角色的工作方式？
- 在 Amazon EC2 中使用角色所需的权限
- 怎样入门？
- 相关信息

Amazon EC2 实例的角色的工作方式？

在下图中，一名开发人员在 Amazon EC2 实例上运行一个应用程序，该应用程序要求访问名为 `amzn-s3-demo-bucket-photos` 的 S3 存储桶。管理员创建 `Get-pics` 服务角色并将该角色附加到 Amazon EC2 实例。该角色包括一个权限策略，该策略授予对指定 S3 存储桶的只读访问权限。它还包括一个信任策略，该策略允许 Amazon EC2 实例担任该角色并检索临时凭证。在该实例上运行应用程序时，应用程序可以使用该角色的临时凭证访问照片存储桶。管理员不必向开发人员授予访问照片存储桶的权限，开发人员完全不必共享或管理凭证。



1. 管理员使用 IAM 创建 **Get-pics** 角色。在角色的信任策略中，管理员指定仅 Amazon EC2 实例能够担任该角色。在角色的权限策略中，管理员为 amzn-s3-demo-bucket-photos 存储桶指定只读权限。
2. 开发人员启动 Amazon EC2 实例，并向该实例分配 Get-pics 角色。

   > **Note**
   >
   > 如果您使用 IAM 控制台，则会为您管理实例配置文件，该配置文件大部分对您是透明的。但是，如果您使用 Amazon CLI 或 API 创建和管理角色和 Amazon EC2 实例，则必须创建实例配置文件并采用单独步骤将角色分配给它。随后在启动实例时，您必须指定实例配置文件名称而不是角色名称。

3. 应用程序在运行时会从 Amazon EC2 实例元数据获取临时安全凭证，如从实例元数据检索安全凭证中所述。这些是临时安全凭证，用于表示角色，在有限时间段内有效。

   使用某些 Amazon 开发工具包，开发人员可以使用提供程序以透明方式管理临时安全凭证。（各 Amazon 开发工具包的文档介绍该软件开发工具包支持的凭证管理功能。）

   应用程序也可以直接从 Amazon EC2 实例的实例元数据中获取临时凭证。凭证和相关值可从元数据的 iam/security-credentials/*role-name* 类别（在本例中为 iam/security-credentials/Get-pics）获得。如果应用程序从实例元数据获取凭证，则它可对凭证进行缓存。
4. 通过使用检索到的临时凭证，应用程序可以访问照片存储桶。由于附加到 **Get-pics** 角色的策略，应用程序具有只读权限。

   实例上提供的临时安全凭证会在过期之前自动更新，因此始终具有有效的凭证集。应用程序只需要确保在当前凭证过期之前从实例元数据获取新的凭证集。可以使用 Amazon 开发工具包管理凭证，这样应用程序就不需要包含额外的逻辑来刷新凭证。例如，使用实例配置文件凭证提供程序实例化客户端。但是，如果应用程序从实例元数据获取临时安全凭证并对它们进行缓存，则它应在当前凭证集过期之前，每隔一小时 (或至少每隔 15 分钟) 获取刷新的凭证集。在 iam/security-credentials/*role-name* 类别中返回的信息中包含过期时间。

在 Amazon EC2 中使用角色所需的权限

要启动具有角色的实例，开发人员必须有启动 Amazon EC2 实例的权限和传递 IAM 角色的权限。

以下示例策略可以实现使用 Amazon Web Services 管理控制台发布引用角色的实例。策略包含通配符 (*) 以允许用户传递任意角色并执行所有列出的 Amazon EC2 操作。用户可以通过 `ListInstanceProfiles` 操作浏览 Amazon Web Services 账户 中的所有角色。

Example授权用户使用 Amazon EC2 控制台启动具有任何角色的实例的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "IamPassRole",
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "ec2.amazonaws.com"
                }
            }
        },
        {
            "Sid": "ListEc2AndListInstanceProfiles",
            "Effect": "Allow",
            "Action": [
                "iam:ListInstanceProfiles",
                "ec2:Describe*",
                "ec2:Search*",
                "ec2:Get*"
            ],
            "Resource": "*"
        }
    ]
}
```

限制哪些角色能够传递到 Amazon EC2 实例（使用 PassRole）

您可以使用 `PassRole` 权限来限制用户启动实例时可以传递给 Amazon EC2 实例的角色。这有助于防止用户运行具有超出用户获授权限的应用程序 即防止用户获取提升的权限。例如，假设用户 Alice 仅有

权启动 Amazon EC2 实例并使用 Amazon S3 存储桶，但是她传递给 Amazon EC2 实例的角色有权使用 IAM 和 Amazon DynamoDB。在这种情况下，Alice 或许能够启动实例，登录它，获取临时安全凭证，然后执行未向她授权的 IAM 或 DynamoDB 操作。

要限制用户可以传递给 Amazon EC2 实例的角色，您可创建一个允许执行 `PassRole` 操作的策略。然后，将策略附加到将启动 Amazon EC2 实例的用户（或该用户所属的 IAM 组）。在策略的 `Resource` 元素中，列出允许用户传递给 Amazon EC2 实例的角色。用户启动一个实例并将角色与该实例关联时，Amazon EC2 会检查是否允许用户传递该角色。当然，您还应确保用户可传递的角色不包含用户不应拥有的权限。

> **Note**
>
> 就像 `PassRole` 或 `RunInstances` 那样，`ListInstanceProfiles` 不是 API 操作。相反，它是只要有角色 ARN 作为参数传递给 API 时，Amazon 便会检查的权限 (或者控制台代表用户这么做)。它帮助管理员控制哪些角色可以由哪些用户传递。在这种情况下，它可确保用户可以将特定角色附加到 Amazon EC2 实例。

Example通过特定角色向用户授予启动 Amazon EC2 实例的权限的示例策略

通过以下示例策略，用户可以使用 Amazon EC2 API，通过角色来启动实例。`Resource` 元素指定角色的 Amazon 资源名称 (ARN)。通过指定 ARN，该策略授予用户仅传递 `Get-pics` 角色的权限。如果在启动实例时用户尝试指定其他角色，则该操作会失败。用户拥有运行任何实例的权限，无论他们是否传递角色。

JSON

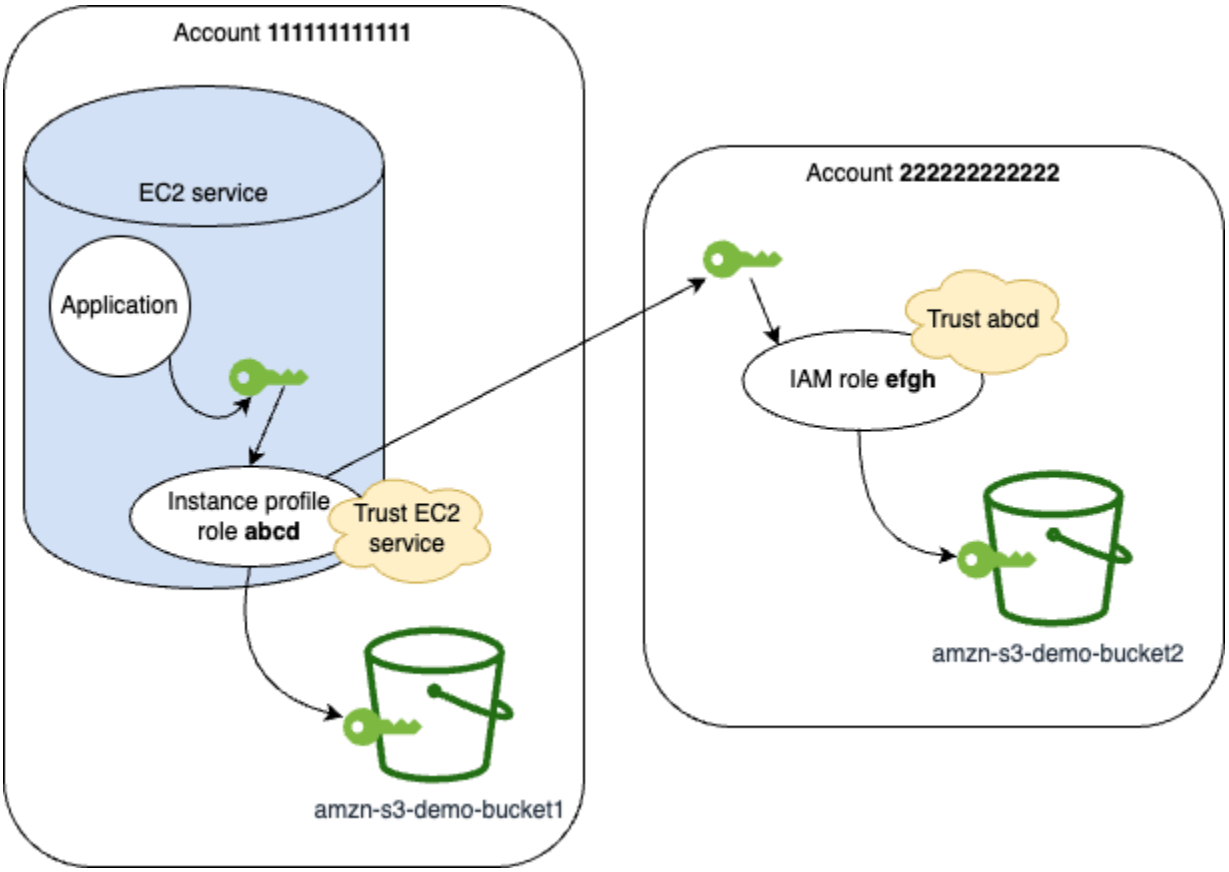
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "arn:aws:iam::111122223333:role/Get-pics"
```

```
        }
    ]
}
```

允许实例配置文件角色切换到另一个账户中的角色

您可以允许 Amazon EC2 实例上运行的应用程序在另一个账户中运行命令。为此，您必须允许第一个账户中的 Amazon EC2 实例角色切换到第二个账户中的角色。

假设您使用两个 Amazon Web Services 账户，并希望允许 Amazon EC2 实例上运行的应用程序在两个账户中运行 Amazon CLI 命令。假设 Amazon EC2 实例位于 111111111111 账户中。该实例包含 abcd 实例配置文件角色，以允许应用程序对同一 111111111111 账户中的 amzn-s3-demo-bucket1 存储桶执行只读 Amazon S3 任务。不过，还必须允许应用程序担任 efgh 跨账户角色以访问 222222222222 账户中的 amzn-s3-demo-bucket2 Amazon S3 存储桶。



abcd Amazon EC2 实例配置文件角色必须具有以下权限策略，以允许应用程序访问 amzn-s3-demo-bucket1 Amazon S3 存储桶：

账户 111111111111 ***abcd*** 角色权限策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1/*",
                "arn:aws:s3:::amzn-s3-demo-bucket1"
            ]
        },
        {
            "Sid": "AllowIPToAssumeCrossAccountRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "arn:aws:iam::222222222222:role/efgh"
        }
    ]
}
```

abcd 角色必须信任 Amazon EC2 服务以担任该角色。为此，abcd 角色必须具有以下信任策略：

账户 111111111111 ***abcd*** 角色信任策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "abcdTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"Service": "ec2.amazonaws.com"}
        }
    ]
}
```

假设 efgh 跨账户角色允许对同一 222222222222 账户中的 amzn-s3-demo-bucket2 存储桶执行只读 Amazon S3 任务。为此，efgh 跨账户角色必须具有以下权限策略：

账户 222222222222 ***efgh*** 角色权限策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAccountLevelS3Actions",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowListAndReadS3ActionOnMyBucket",
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
```

```
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket2/*",
                "arn:aws:s3:::amzn-s3-demo-bucket2"
            ]
        }
    ]
}
```

efgh 角色必须信任 abcd 实例配置文件角色以担任该角色。为此，efgh 角色必须具有以下信任策略：

账户 222222222222 ***efgh*** 角色信任策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "efghTrustPolicy",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111111111111:role/abcd"}
        }
    ]
}
```

怎样入门？

为了理解角色如何在 Amazon EC2 实例中起作用，您需要使用 IAM 控制台创建一个角色，启动使用该角色的 Amazon EC2 实例，然后检查正在运行的实例。您可以检查实例元数据，以了解如何为实例提供角色的临时凭证。也可以了解实例上运行的应用程序如何使用角色。使用以下资源了解更多信息。

-
- SDK 演示。Amazon 开发工具包文档包含一些演示，这些演示说明了在 Amazon EC2 实例上运行的应用程序使用角色的临时凭证读取 Amazon S3 存储桶。以下每个演示都采用了不同的编程语言来呈现类似的步骤：

- 适用于 Java 的 Amazon SDK 开发人员指南中的使用 SDK for Java 为 Amazon EC2 配置 IAM 角色
- 《适用于 .NET 的 Amazon SDK 开发人员指南》中的使用 SDK for .NET 启动 Amazon EC2 实例
- 适用于 Ruby 的 Amazon SDK 开发人员指南中的使用 SDK for Ruby 创建 Amazon EC2 实例

相关信息

有关创建角色或适用于 Amazon EC2 实例的角色的更多信息，请参阅以下信息：

- 有关将 IAM 角色与 Amazon EC2 实例结合使用的更多信息，请转至《Amazon EC2 用户指南》。
- 如需创建角色，请参阅IAM 角色创建
- 有关使用临时安全凭证的更多信息，请参阅IAM 临时安全凭证。
- 如果您操作 IAM API 或 CLI，必须创建并管理 IAM 实例配置文件。有关实例配置文件的更多信息，请参阅使用实例配置文件。
- 有关实例元数据中的角色临时安全凭证的更多信息，请参阅《Amazon EC2 用户指南》中的通过实例元数据检索安全凭证。

使用实例配置文件

使用实例配置文件将 IAM 角色传递给 EC2 实例。有关更多信息，请参阅《Amazon EC2 用户指南》中的适用于 Amazon EC2 的 IAM 角色。

管理实例配置文件（控制台）

如果使用 Amazon Web Services 管理控制台 创建 Amazon EC2 的角色，则控制台自动创建实例配置文件，将其命名为与角色相同的名称。当您随后通过 IAM 角色使用 Amazon EC2 控制台启动实例时，可以选择一个角色与该实例关联。在控制台中，显示的列表实际上是实例配置文件名称的列表。控制台不会为不与 Amazon EC2 关联的角色创建实例配置文件。

如果该角色和实例配置文件的名称相同，则您可以使用 Amazon Web Services 管理控制台 删除 Amazon EC2 的 IAM 角色和实例配置文件。要了解删除实例配置文件的更多信息，请参阅 删除角色或实例配置文件。

> **Note**
>
> 要更新实例的权限，请替换其实例配置文件。我们不建议从实例配置文件中移除角色，因为此更改最长需要一小时才会生效。

管理实例配置文件（Amazon CLI 或 Amazon API）

如果您从 Amazon CLI 或 Amazon API 管理角色，则需要使用单独的操作创建角色和实例配置文件。由于角色和实例配置文件可能具有不同名称，则您必须知道实例配置文件的名称及其包含的角色名称。这样，您才能在启动 EC2 实例时选择正确的实例配置文件。

您可以将标签附加到 IAM 资源（包括实例配置文件），以识别、组织和控制对这些资源的访问。只有在使用 Amazon CLI 或 Amazon API 时，才能标记实例配置文件。

> **Note**
>
> 一个实例配置文件只能包含一个 IAM 角色，不过一个角色可以包含在多个实例配置文件中。不能提高每个实例配置文件一个角色这一限制。您可以删除实例配置文件中的现有角色，然后添加另一角色。由于最终一致性，您必须等待此更改在整个 Amazon 中出现。要强制进行更改，您必须取消关联实例配置文件，然后再次关联实例配置文件；您也可以停止并重新启动实例。

管理实例配置文件 (Amazon CLI)

您可以使用以下 Amazon CLI 命令在 Amazon 账户中处理实例配置文件。

- 创建实例配置文件: `aws iam create-instance-profile`
- 标记实例配置文件：`aws iam tag-instance-profile`
- 列出实例配置文件的标签：`aws iam list-instance-profile-tags`
- 取消标记实例配置文件：`aws iam untag-instance-profile`
- 向实例配置文件添加角色: `aws iam add-role-to-instance-profile`
- 列出实例配置文件: `aws iam list-instance-profiles`, `aws iam list-instance-profiles-for-role`
- 获取有关实例配置文件的信息: `aws iam get-instance-profile`
- 从实例配置文件中删除角色: `aws iam remove-role-from-instance-profile`
- 删除实例配置文件: `aws iam delete-instance-profile`

您还可以使用以下命令，将角色附加到已在运行的 EC2 实例。有关更多信息，请参阅 Amazon EC2 的 IAM 角色。

- 将实例配置文件随角色附加到已停止或正在运行的 EC2 实例: `aws ec2 associate-iam-instance-profile`

- 获取 EC2 实例附加的实例配置文件信息: aws ec2 describe-iam-instance-profile-associations
- 将角色的实例配置文件与已停止或正在运行的 EC2 实例分离: aws ec2 disassociate-iam-instance-profile

管理实例配置文件 (Amazon API)

您可以调用以下 Amazon API 操作在 Amazon Web Services 账户 中处理实例配置文件。

- 创建实例配置文件: CreateInstanceProfile
- 标记实例配置文件：TagInstanceProfile
- 列出实例配置文件上的的标签 ：ListInstanceProfileTags
- 取消标记实例配置文件：UntagInstanceProfile
- 向实例配置文件添加角色: AddRoleToInstanceProfile
- 列出实例配置文件: ListInstanceProfiles, ListInstanceProfilesForRole
- 获取有关实例配置文件的信息: GetInstanceProfile
- 从实例配置文件中删除角色: RemoveRoleFromInstanceProfile
- 删除实例配置文件: DeleteInstanceProfile

您还可以调用以下操作，将角色附加到已在运行的 EC2 实例。有关更多信息，请参阅 Amazon EC2 的 IAM 角色。

- 将实例配置文件随角色附加到已停止或正在运行的 EC2 实例: AssociateIamInstanceProfile
- 获取 EC2 实例附加的实例配置文件信息: DescribeIamInstanceProfileAssociations
- 将角色的实例配置文件与已停止或正在运行的 EC2 实例分离: DisassociateIamInstanceProfile

身份提供程序和 Amazon 中的联合身份验证

作为最佳实践，建议您要求人类用户使用与身份提供商的联合身份验证访问 Amazon 资源，而不是在您的 Amazon Web Services 账户 中创建单独的 IAM 用户。利用身份提供程序 (IdP)，您可以管理 Amazon 外部的用户身份，并向这些外部用户身份授予使用您账户中的 Amazon 资源的权限。如果您的组织已有自己的身份系统（如企业用户目录），这将十分有用。如果要创建需要访问 Amazon 资源的移动应用程序或 Web 应用程序，这也十分有用。

> **Note**
>
> 您还可以使用外部 SAML 身份提供商在 IAM Identity Center 中管理人类用户，而不是在 IAM 中使用 SAML 联合身份验证。IAM Identity Center 与身份提供商的联合身份验证使您能够允许人员访问组织中的多个 Amazon 账户和多个 Amazon 应用程序。要了解需要使用 IAM 用户的特定情况，请参阅 何时创建 IAM 用户（而非角色）。

如果您喜欢在不启用 IAM Identity Center 的情况下使用单个 Amazon 账户，则可以将 IAM 与外部 IdP 结合使用，后者会使用 OpenID Connect（OIDC）或 SAML 2.0（安全断言标记语言 2.0）向 Amazon 提供身份信息。OIDC 将不在 Amazon 上运行的应用程序（例如 GitHub Actions）连接到 Amazon 资源。知名的 SAML 身份提供者的示例如 Shibboleth 和 Active Directory 联合身份验证服务。

使用身份提供商时，您不必创建自定义登录代码或管理自己的用户身份。IdP 将向您提供它们。您的外部用户通过 IdP 登录，您可以向这些外部身份授予使用您的账户中的 Amazon 资源的权限。身份提供者可帮助您确保 Amazon Web Services 账户 的安全，因为您不必再在应用程序中分配或嵌入长期安全凭证（如访问密钥）。

查看下表，以帮助确定哪种 IAM 联合身份验证类型最适合您的使用案例：IAM、IAM Identity Center 还是 Amazon Cognito。以下摘要和表格概述了您的用户可以用来获得对 Amazon 资源的联合访问权限的方法。

IAM 联合身份验证类型	账户类型	对……的访问管理	支持的身份源
使用 IAM Identity Center 的联合身份验证	由 Amazon Organizations 管理的多个账户	您的人力的人类用户	• SAML 2.0 • 托管式 Active Directory • Identity Center 目录
使用 IAM 的联合身份验证	单一独立账户	• 短期、小规模部署中的人类用户 • 机器用户	• SAML 2.0 • OIDC
使用 Amazon Cognito 身份池的联合身份验证	任何	需要 IAM 授权才能访问资源的应用程序的用户	• SAML 2.0 • OIDC

IAM 联合身份验证类型	账户类型	对……的访问管理	支持的身份源
			• 选择 OAuth 2.0 社交身份提供者

使用 IAM Identity Center 的联合身份验证

为方便集中管理人类用户的访问权限，我们建议使用 IAM Identity Center 来管理对您账户的访问以及这些账户中的权限。IAM Identity Center 中的用户将被授予对您的 Amazon 资源的长期凭证。您可以使用 Active Directory、外部身份提供者（IdP）或 IAM Identity Center 目录作为身份源，供用户和组分配对您的 Amazon 资源的访问权限。

IAM Identity Center 支持使用 SAML（安全断言标记语言）2.0 的身份联合验证，为被授权在 Amazon 访问门户中使用应用程序的用户提供联合单点登录访问权限。然后，用户可以单点登录到支持 SAML 的服务，包括 Amazon Web Services 管理控制台 和第三方应用程序（如 Microsoft 365、SAP Concur 和 Salesforce）。

使用 IAM 的联合身份验证

虽然我们强烈建议在 IAM Identity Center 中管理人类用户，但在短期、小规模部署中，您可以通过 IAM 为人类用户启用联合主体访问权限。IAM 允许您使用单独的 SAML 2.0 和 Open ID Connect（OIDC）IdP，并使用联合主体属性进行访问控制。借助 IAM，您可以将用户属性（例如成本中心、职务或区域设置）从您的 IdP 传递给 Amazon，并根据这些属性实施精细访问权限。

工作负载是一系列资源和代码，它们可提供商业价值，如应用程序或后端过程。您的工作负载可能需要 IAM 身份才能向 Amazon 服务、应用程序、操作工具和组件发出请求。这些身份包括运行在您的 Amazon 环境（例如 Amazon EC2 实例或 Amazon Lambda 函数）中运行的计算机。

您还可以为需要访问权限的外部团体管理计算机身份。要为计算机身份授予访问权限，您可以使用 IAM 角色。IAM 角色具有特定的权限并可通过使用带有角色会话的临时安全凭证提供访问 Amazon 的方法。此外，您还可以拥有位于 Amazon 以外且需要访问您的 Amazon 环境的计算机。对于在 Amazon 外部运行的计算机，您可以使用 IAM Roles Anywhere。有关角色的更多信息，请参阅 IAM 角色。有关如何使用角色跨 Amazon Web Services 账户 委派访问权限的详情，请参阅 IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限。

要将 IdP 直接连接到 IAM，您需要创建 IAM 身份提供者实体，以在您的 Amazon Web Services 账户 和 IdP 之间建立信任关系。IAM 支持与 OpenID Connect (OIDC) 或者 SAML 2.0 (Security Assertion

Markup Language 2.0) 兼容的 IdPs。有关通过 Amazon 使用这些 IdP 之一的更多信息，请参阅以下部分：

- OIDC 联合身份验证
- SAML 2.0 联合身份验证

使用 Amazon Cognito 身份池的联合身份验证

Amazon Cognito 专为想要在其移动应用程序和 Web 应用程序中对用户进行身份验证和授权的开发人员而设计。Amazon Cognito 用户池为您的应用程序添加登录和注册功能，身份池提供 IAM 凭证，授予您的用户访问您在 Amazon 中管理的受保护资源的权利。身份池通过 `AssumeRoleWithWebIdentity` API 操作获取临时会话的凭证。

Amazon Cognito 与支持 SAML 和 OpenID Connect 的外部身份提供者以及 Facebook、Google 和 Amazon 等社交身份提供者合作。您的应用程序可以使用用户群体或外部 IdP 登录用户，然后在 IAM 角色中使用自定义的临时会话代表他们检索资源。

其他资源

- 有关如何创建自定义联合身份验证代理以使用组织的身份验证系统实现到 Amazon Web Services 管理控制台 的单点登录（SSO）的演示，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。

常见场景

> **Note**
>
> 我们建议您要求您的人类用户在访问 Amazon 时使用临时凭证。您是否考虑过使用 Amazon IAM Identity Center？ 您可以使用 IAM Identity Center 集中管理对多个 Amazon Web Services 账户 的访问权限，并为用户提供受 MFA 保护的单点登录访问权限，可从一个位置访问其分配的所有账户。借助 IAM Identity Center，您可以在 IAM Identity Center 中创建和管理用户身份，或者轻松连接到现有的 SAML 2.0 兼容身份提供者。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

您可以使用外部身份提供者 (IdP) 来管理 Amazon 之外的用户身份以及外部 IdP。外部 IdP 可以使用 OpenID Connect（OIDC）或安全断言标记语言（SAML）向 Amazon 提供身份信息。OIDC 通常在不在 Amazon 上运行的应用程序需要访问 Amazon 资源时使用。

如果要使用外部 IdP 配置联合身份验证，可以创建 IAM 身份提供商，以将外部 IdP 及其配置告知 Amazon。这样将在您的 Amazon Web Services 账户和外部 IdP 之间建立信任。以下主题提供了使用 IAM 身份提供者的常见场景。

主题

- 用于移动应用程序的 Amazon Cognito
- 移动应用程序的 OIDC 联合身份验证

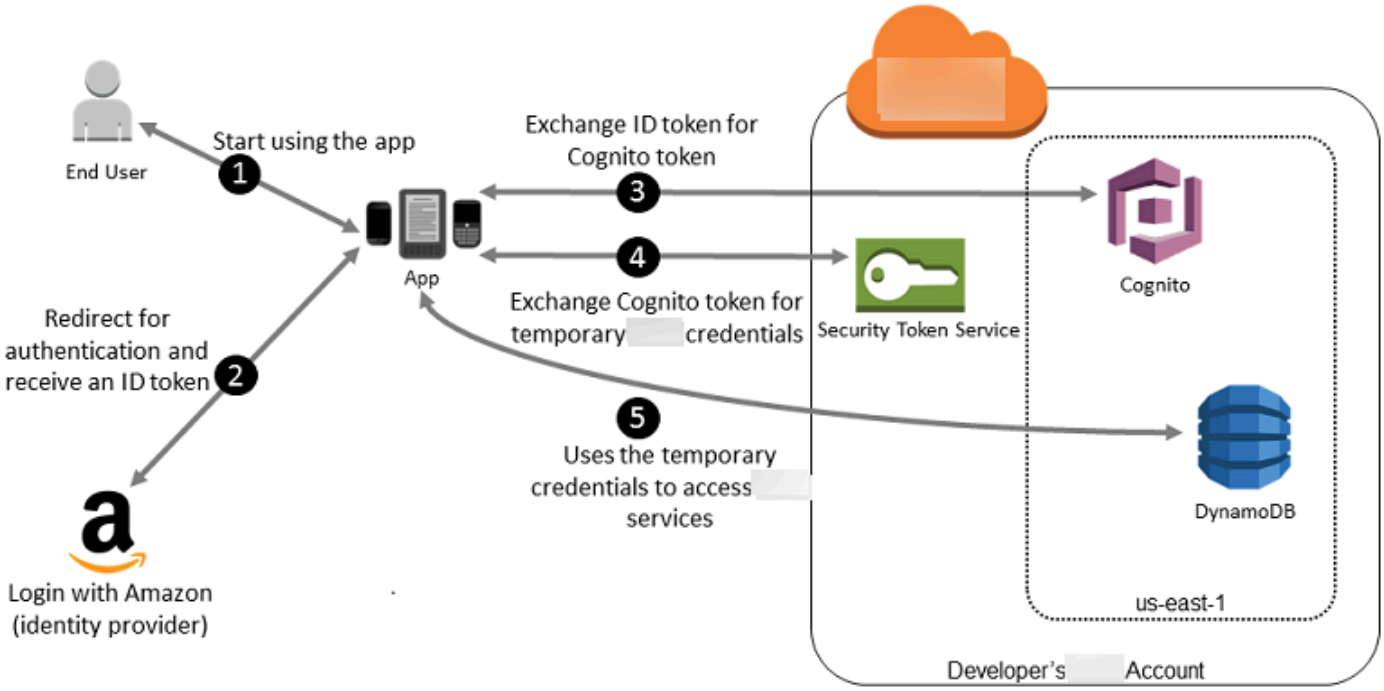
用于移动应用程序的 Amazon Cognito

使用 OIDC 联合身份验证的首选方式是使用 Amazon Cognito。例如，开发人员 Adele 正在制作一款用于移动设备的游戏，其中将分数和个人资料等用户数据存储在 Amazon S3 和 Amazon DynamoDB 中。Adele 还可将此数据存储在本地设备上，并使用 Amazon Cognito 来跨设备保持数据的同步。她知道，出于安全性和维护的原因，不应随游戏分配长期 Amazon 安全凭证。她还知道，这个游戏可能有大量用户。出于所有这些原因，她不想在 IAM 中为每个玩家都新建用户身份，而是将游戏制作成用户可使用其已通过知名的外部身份提供者 (IdP)（如 Login with Amazon、Facebook、Google 或任何 OpenID Connect (OIDC) 兼容的 IdP）建立的身份进行登录。她的游戏可利用其中某个提供商的身份验证机制验证用户的身份。

为使该移动应用程序可访问她的 Amazon 资源，Adele 首先向她选择的 IdP 注册一个开发人员 ID。她还按其中某个提供商的要求配置该应用程序。在她的包含该游戏的 Amazon S3 存储桶和 DynamoDB 表的 Amazon Web Services 账户 中，Adele 使用 Amazon Cognito 创建了精确定义该游戏所需权限的 IAM 角色。如果她使用的是 OIDC IdP，她还可创建 IAM OIDC 身份提供商实体以在其 Amazon Web Services 账户 中的 Amazon Cognito 身份池和该 IdP 之间建立信任关系。

在应用程序的代码中，Adele 调用其先前配置的 IdP 的登录接口。IdP 处理让用户登录的所有详细信息，然后应用程序从提供商那里获得 OAuth 访问令牌或 OIDC ID 令牌。Adele 的应用程序可使用此身份验证信息换取一组临时安全凭证 (包括 Amazon 访问密钥 ID、秘密访问密钥和会话令牌)。然后，应用程序可以使用这些凭证访问 Amazon 提供的 Web 服务。该应用程序仅获得在其担任的角色中定义的权限。

下图以 Login with Amazon 作为 IdP，展示此过程运行方式的简化流程。对于步骤 2，该应用程序还可以使用 Facebook、Google 或任何与 OIDC 兼容的 IdP，但此处不进行说明。



1. 客户在移动设备上启动您的应用程序。应用程序要求用户登录。
2. 应用程序使用 Login with Amazon 资源接受用户凭证。
3. 应用程序使用 Amazon Cognito API 操作 `GetId` 和 `GetCredentialsForIdentity` 将 Login with Amazon ID 令牌交换成 Amazon Cognito 令牌。Amazon Cognito 已配置为信任 Login with Amazon 项目，它会生成一个令牌，用于与 Amazon STS 交换临时会话凭证。
4. 该应用程序从 Amazon Cognito 接收临时安全凭证。您的应用程序还可使用 Amazon Cognito 中的基本（经典）工作流程，通过 `AssumeRoleWithWebIdentity` 从 Amazon STS 中检索令牌。有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的身份池（联合身份）身份验证流程。
5. 应用程序可使用临时安全凭证访问应用程序运行所需的任何 Amazon 资源。与临时安全凭证关联的角色及其分配的策略将决定可访问的资源。

使用以下过程将您的应用程序配置为使用 Amazon Cognito 对用户进行身份验证，并向您的应用程序授予对 Amazon 资源的访问权限。有关完成此操作的具体步骤，请参阅 Amazon Cognito 的相关文档。

1. （可选）使用 Login with Amazon、Facebook、Google 或任何其他与 OpenID Connect (OIDC) 兼容的 IdP 注册为开发人员，并使用这些提供商配置一个或多个应用程序。该步骤是可选的，因为 Amazon Cognito 还支持您的用户进行未经身份验证的（来宾）访问。
2. 转到 Amazon Cognito 控制台 Amazon Web Services 管理控制台。使用 Amazon Cognito 向导创建一个身份池，身份池是一种供 Amazon Cognito 保留为您的应用程序组织的终端用户身份的容

器。您可以在应用程序间共享身份池。在设置身份池时，Amazon Cognito 将创建一个或两个定义 Amazon Cognito 用户的权限的 IAM 角色（一个角色针对经过身份验证的身份，另一个角色针对未经身份验证的“来宾”身份）。

3. 将 Amazon Amplify 与您的应用程序集成，然后导入使用 Amazon Cognito 所需的文件。
4. 创建一个 Amazon Cognito 凭证提供程序实例，并传递身份池 ID、您的 Amazon Web Services 账户 账号，以及与身份池关联的角色的 Amazon 资源名称（ARN）。Amazon Web Services 管理控制台 中的 Amazon Cognito 向导提供了可帮助您入门的示例代码。
5. 当您的应用程序访问 Amazon 资源时，可将该凭证提供程序实例传递给客户端对象，从而将临时安全凭证传递给该客户端。证书的权限基于您之前定义的角色。

有关更多信息，请参阅下列内容：

- Amazon Amplify 框架文档中的登录（Android）。
- Amazon Amplify 框架文档中的登录（iOS）。

移动应用程序的 OIDC 联合身份验证

为获得最佳效果，请在几乎所有 OIDC 联合身份验证场景中将 Amazon Cognito 用作身份凭证代理程序。Amazon Cognito 易于使用，并提供了额外功能，如匿名（未经身份验证的）访问，以及跨设备和提供商同步用户数据。但是，如果您已通过手动调用 `AssumeRoleWithWebIdentity` API 创建了使用 OIDC 联合身份验证的应用程序，也可继续使用它，您的应用程序仍能正常工作。

在不 使用 Amazon Cognito 的情况下使用 OIDC 联合身份验证的过程遵循此概要：

1. 以开发人员身份注册到外部身份提供者 (IdP)，然后使用向您提供应用程序的唯一 ID 的 IdP 来配置您的应用程序。（不同的 IdP 使用不同的术语表示此过程。此概要使用配置一词表示通过 IdP 标识应用程序的过程。）每个 IdP 均提供一个对于该 IdP 独一无二的应用程序 ID，因此如果按多个 IdP 的要求配置同一应用程序，则该应用程序将有多个应用程序 ID。可按每个提供商的要求配置多个应用程序。

   以下外部链接提供有关使用一些常用身份提供者 (IdP) 的信息：

   - Login with Amazon 开发人员中心
   - 在 Facebook 开发人员网站上，将 Facebook 登录名添加到您的应用程序或网站。
   - 在 Google 开发人员网站上，使用 OAuth 2.0 进行登录 (OpenID Connect)。

> **Important**
>
> 如果您使用 Google、Facebook 或 Amazon Cognito 提供的 OIDC 身份提供程序，请勿在 Amazon Web Services 管理控制台 中创建单独的 IAM 身份提供者，Amazon 内置了这些 OIDC 身份提供程序，可供您使用。跳过以下步骤，直接使用身份提供者创建新角色。

2. 如果您使用与 OIDC 兼容、除 Google、Facebook 或 Amazon Cognito 以外的 IdP，请为其创建 IAM 身份提供者实体。
3. 在 IAM 中，您可以创建一个或多个角色。对于每个角色，定义谁可代入该角色（信任策略）和应用程序的用户将具有什么权限（权限策略）。通常，您为应用程序支持的每个 IdP 创建一个角色。例如，可创建一个在用户通过 Login with Amazon 登录时应用程序代入的角色，为同一应用程序再创建第二个角色，其中用户通过 Facebook 登录，然后为该应用程序创建第三个角色，其中用户通过 Google 登录。对于信任关系，指定 IdP (如 Amazon.com) 作为 `Principal` (可信实体)，并加入一个与 IdP 分配的应用程序 ID 匹配的 `Condition`。为第三方身份提供者创建角色 将介绍不同提供商的角色示例。
4. 在应用程序中，通过 IdP 验证用户身份。执行此操作的方式的详情因您所使用的 IdP (Login with Amazon、Facebook 或 Google) 和运行应用程序的平台而异。例如，Android 应用程序的身份验证方式与 iOS 应用程序或基于 JavaScript 的 Web 应用程序的不同。

   通常，如果用户尚未登录，则 IdP 负责显示登录页面。Idp 在对用户进行身份验证后，会将身份验证令牌与用户相关信息一起返回到您的应用程序。包含的信息取决于 IdP 公开的内容和用户愿意共享的信息。可在应用程序中使用这些信息。
5. 在应用程序中，对 操作进行未签名 `AssumeRoleWithWebIdentity` 调用以请求临时安全凭证。在该请求中，传递 IdP 的身份验证令牌，然后指定为该 IdP 创建的 IAM 角色的 Amazon 资源名称 (ARN)。Amazon 将验证令牌是否可信和有效，如果是这样，则会将临时安全凭证返回到具有从您在请求中命名的角色的权限的应用程序。响应中还包括来自 IdP 的用户相关元数据，例如，IdP 将其与用户关联的唯一用户 ID。
6. 通过使用来自 `AssumeRoleWithWebIdentity` 响应的临时安全凭证，应用程序向 Amazon API 操作发出已签名的请求。IdP 中的用户 ID 信息可以区分您的应用程序中的用户。例如，您可以将对象放入 Amazon S3 文件夹中，其包含用户 ID 作为前缀或后缀。这样可创建锁定该文件夹的访问控制策略，以使仅具有该 ID 的用户能够访问该文件夹。有关更多信息，请参阅 Amazon STS 联合用户主体。
7. 您的应用程序应缓存临时安全凭证，这样您就不需要每次在应用程序需要对 Amazon 发出请求时获取新凭证。默认情况下，证书的有效期为 1 小时。当凭证到期时 (或在此之前)，再次调用 `AssumeRoleWithWebIdentity` 以获取新的一组临时安全凭证。根据 IdP 及其管理其令牌的方

式，可能必须先刷新 IdP 的令牌，然后再对 `AssumeRoleWithWebIdentity` 进行新的调用，因为 IdP 的令牌通常也在固定的一定时间后到期。如果使用的是 Amazon SDK for iOS 或 Amazon SDK for Android，则可使用 AmazonSTSCredentialsProvider 操作，该操作管理 IAM 临时凭证，包括按需刷新凭证。

OIDC 联合身份验证

假设您要创建一个访问 Amazon 资源的应用程序，例如使用工作流访问 Amazon S3 和 DynamoDB 的 GitHub Actions。

当您使用这些工作流时，将向必须使用 Amazon 访问密钥进行签名的 Amazon 服务提出请求。但是，我们强烈建议您不要将 Amazon 凭证长期存储在 Amazon 之外的应用程序中。相反，使用 OIDC 联合身份验证将应用程序配置为在需要时动态请求临时 Amazon 安全凭证。提供的临时凭证会映射到一个 Amazon 角色，该角色将只拥有执行该应用程序所需任务的必要权限。

借助 OIDC 联合身份验证，您不需要创建自定义登录代码或管理自己的用户身份。相反，您可以在应用程序（例如 GitHub Actions 或任何其他兼容 OpenID Connect（OIDC）的 IdP）中使用 OIDC 进行 Amazon 身份验证。他们会接收身份验证令牌（称为 JSON Web 令牌，JWT），然后用该令牌交换 Amazon 中的临时安全凭证，这些凭证映射到有权使用您 Amazon Web Services 账户 中资源的 IAM 角色。使用 IdP 有助您确保 Amazon Web Services 账户的安全，因为您不必随应用程序嵌入和分配长期安全凭证。

OIDC 联合身份验证支持机器对机器身份验证（例如，CI/CD 管道、自动脚本和无服务器应用程序）和人工用户身份验证。对于需要管理用户注册、登录和用户个人资料的人工用户身份验证场景，可以考虑将 Amazon Cognito 用作身份凭证代理程序。有关将 Amazon Cognito 与 OIDC 结合使用的详细信息，请参阅 用于移动应用程序的 Amazon Cognito。

> **Note**
>
> 由 OpenID Connect (OIDC) 身份提供程序颁发的 JSON Web 令牌 (JWT) 在 `exp` 声明中包含指定令牌何时过期的过期时间。在 OpenID Connect (OIDC) Core 1.0 标准允许的情况下，IAM 在 JWT 中指定的到期时间之外提供五分钟的时段来解决时钟偏差。这意味着将接受 IAM 在到期时间后但在这五分钟内收到的 OIDC JWT 以进行进一步的评估和处理。

主题

- 有关 OIDC 联合身份验证的其他资源

- 在 IAM 中创建 OpenID Connect（OIDC）身份提供者
- 获取 OpenID Connect 身份提供者的指纹
- 适用于共享 OIDC 提供者的身份提供者控制

有关 OIDC 联合身份验证的其他资源

以下资源可帮助您详细了解 OIDC 联合身份验证：

- 通过在 Amazon Web Services 中配置 OpenID Connect，从而在 GitHub 工作流中使用 OpenID Connect
- 《适用于 Android 的 Amplify 库指南》中的 Amazon Cognito 身份和《适用于 Swift 的 Amplify 库指南》中的 Amazon Cognito 身份。
- Amazon 安全博客上的授予 Amazon 资源访问权限时如何使用外部 ID提供了安全配置跨账户访问和外部身份联合验证的指导。

在 IAM 中创建 OpenID Connect（OIDC）身份提供者

IAM OIDC 身份提供程序是 IAM 中的实体，这些实体描述支持 OpenID Connect (OIDC) 标准的身份提供程序 (IdP) 服务，如 Google 或 Salesforce）。当您要在与 OIDC 兼容的 IdP 和您的 Amazon Web Services 账户 之间建立信任时，请使用 IAM OIDC 身份提供程序。如果您正在创建需要访问 Amazon 资源的移动应用或 Web 应用程序，但又不想创建自定义登录代码或管理您自己的用户身份，这会很有用。有关此方案的更多信息，请参阅the section called “OIDC 联合身份验证”。

您可以使用 Amazon Web Services 管理控制台、Amazon Command Line Interface、Tools for Windows PowerShell 或 IAM API 创建和管理 IAM OIDC 身份提供程序。

创建 IAM OIDC 身份提供程序后，必须创建一个或多个 IAM 角色。角色是 Amazon 中的一个实体，它没有自己的凭证（与用户一样）。但在此上下文中，角色将动态分配给由组织的 IdP 验证的 OIDC 联合主体。该角色允许组织的 IdP 请求临时安全凭证以便访问 Amazon。分配给该角色的策略决定了用户可在 Amazon 中执行的操作。要为第三方身份提供程序创建角色，请参阅 为第三方身份提供者创建角色。

> **Important**
>
> 为支持 `oidc-provider` 资源的操作配置基于身份的策略时，IAM 会评估完整的 OIDC 身份提供者 URL，包括任何指定的路径。如果 OIDC 身份提供者 URL 包含路径，则必须将该路径作为 `Resource` 元素值包含在 `oidc-provider` ARN 中。您还可以选择将正斜杠和通配符

（/*）附加到 URL 域，或者在 URL 路径中的任何位置使用通配符（* 和？）。如果请求中的 OIDC 身份提供者 URL 与策略中 Resource 元素设置的值不匹配，请求将会失败。

要解决有关 IAM OIDC 联合身份验证的常见问题，请参阅 Amazon re:Post 上的解决与 OIDC 相关的错误。

主题

- 先决条件：验证身份提供商的配置
- 创建和管理 OIDC 提供商（控制台）
- 创建和管理 IAM OIDC 身份提供程序 (Amazon CLI)
- 创建和管理 OIDC 身份提供程序 (Amazon API)

先决条件：验证身份提供商的配置

在创建 IAM OIDC 身份提供商之前，您必须从 IdP 处获得以下信息。有关获取 OIDC 提供商配置信息的更多信息，请参阅 IdP 的文档。

1. 确定您的 OIDC 身份提供商的公开可用 URL。该 URL 必须以 https:// 开头。根据 OIDC 标准，允许使用路径组件，但不允许使用查询参数。通常，该 URL 只包含一个主机名，如 https://server.example.org 或 https://example.com。URL 不应包含端口号。
2. 在 OIDC 身份提供商的 URL 末尾添加 /.well-known/openid-configuration，以查看该提供商的公开可用配置文档和元数据。您必须有一个 JSON 格式的发现文档，其中包含提供商的配置文档和元数据，这些文档和元数据可以从 OpenID Connect 提供商发现端点 URL 中检索。
3. 确认以下值包含在提供商的配置信息中。如果 openid 配置缺少这些字段中的任何一个，则必须更新发现文档。此过程可能因身份提供商而异，因此请按照 IdP 文档完成此任务。

    - issuer：域的 URL。
    - jwks_uri：IAM 获取公钥的 JSON Web 密钥集 (JWKS) 端点。您的身份提供商必须在 openid 配置中包含一个 JSON Web 密钥集 (JWKS) 端点。此 URI 定义了从何处获取用于验证来自身份提供商的签名令牌的公钥。

    > **Note**
    >
    > JSON Web 密钥集 (JWKS) 必须至少包含一个密钥，最多可以有 100 个 RSA 密钥和 100 个 EC 密钥。如果 OIDC 身份提供商的 JWKS 中包含超过 100 个 RSA 密钥或 100 个 EC 密钥，则在搭配使用 AssumeRolewithEbidentity API 操作以及由超过 100 个密钥

限制的密钥类型签名的 JWT 时，将返回 `InvalidIdentityToken` 异常。例如，如果 JWT 使用 RSA 算法进行签名，而提供商的 JWKS 中含有超过 100 个 RSA 密钥，则会返回 `InvalidIdentityToken` 异常。

- claims_supported：有关用户的信息，可帮助确保来自 IdP 的 OIDC 身份验证响应包含 IAM 策略中 Amazon 用于检查 OIDC 联合主体的必需属性。有关可用于声明的 IAM 条件键的列表，请参阅 Amazon OIDC 联合身份验证的可用键。
  - aud：必须在 JSON Web 令牌 (JWT) 中确定 IdP 发布的受众声明值。受众 (aud) 声明是特定于应用程序的，用于标识令牌的预期接收者。当您向 OpenID Connect 提供商注册移动或 Web 应用时，他们会建立一个客户端 ID 来标识该应用程序。客户端 ID 是应用的唯一标识符，在 aud 声明中传递用于进行身份验证。在创建 IAM OIDC 身份提供商时，aud 声明必须与“受众”值匹配。
  - iat：声明必须包含 `iat` 的值，该值表示 ID 令牌的发布时间。
  - iss：身份提供商的 URL。URL 必须以 https:// 开头，并且应与提供给 IAM 的提供商 URL 相对应。根据 OIDC 标准，允许使用路径组件，但不允许使用查询参数。通常，该 URL 只包含一个主机名，如 https://server.example.org 或 https://example.com。URL 不应包含端口号。
- response_types_supported：id_token
- subject_types_supported：public
- id_tokeninging_alg_values_suirted：RS256、RS384、RS512、ES256、ES384、ES512

> **Note**
>
> 可以在下面的示例中包括其他声明，如 `my_custom_claim`；但是，Amazon STS 将忽略该声明。

```
{
  "issuer": "https://example-domain.com",
  "jwks_uri": "https://example-domain.com/jwks/keys",
  "claims_supported": [
    "aud",
    "iat",
    "iss",
    "name",
    "sub",
    "my_custom_claim"
  ],
  "response_types_supported": [
```

```
    "id_token"
  ],
  "id_token_signing_alg_values_supported": [
    "RS256",
    "RS384",
    "RS512",
    "ES256",
    "ES384",
    "ES512"
  ],
  "subject_types_supported": [
    "public"
  ]
}
```

创建和管理 OIDC 提供商（控制台）

按照以下说明在 Amazon Web Services 管理控制台 中创建和管理 IAM OIDC 身份提供程序。

> **Important**
>
> 如果您使用的是 Google、Facebook 或 Amazon Cognito 的 OIDC 身份提供程序，请勿使用此过程创建单独的 IAM 身份提供程序。这些 OIDC 身份提供程序已经内置到 Amazon，并可供您使用。相反，请按照以下步骤为身份提供程序创建新角色，请参阅 创建用于 OpenID Connect 联合身份验证（控制台）的角色。

创建 IAM OIDC 身份提供程序（控制台）

1. 在创建 IAM OIDC 身份提供程序之前，您必须向 IdP 注册您的应用程序以便接收客户端 ID。客户端 ID（也称为受众）是您的应用的唯一标识符，在您向 IdP 注册您的应用时颁发给您。有关如何获取客户端 ID 的更多信息，请参阅您的 IdP 的文档。

   > **Note**
   >
   > Amazon 使用我们的受信任根证书颁发机构（CA）库来保护与 OIDC 身份提供者（IdP）之间的通信，从而验证JSON Web 密钥集（JWKS）端点的 TLS 证书。如果您的 OIDC IdP 依赖的证书不是由其中某个受信任的 CA 签名，则仅在此时我们会使用 IdP 配置中设

置的指纹来保护通信。如果我们无法检索 TLS 证书或需要 TLS v1.3，则 Amazon 将回退到指纹验证。

2. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 在导航窗格中，选择 Identity providers（身份提供程序），然后选择 Add provider（添加提供商）。
4. 对于 Configure provider（配置提供商），选择 OpenID Connect。
5. 对于 Provider URL，键入 IdP 的 URL。该 URL 必须遵从以下限制：
   - 该 URL 区分大小写。
   - URL 必须以 **`https://`** 开头。
   - URL 不应包含端口号。
   - 在您的 Amazon Web Services 账户 中，每个 IAM OIDC 身份提供程序都必须有唯一的 URL。如果尝试提交的 URL 已用于 Amazon Web Services 账户 中的 OpenID Connect 提供商，则会出现错误。
6. 对于 Audience，键入您向 IdP 注册的、在Step 1 中接收的并且将向 Amazon 发出请求的应用程序的客户端 ID。如果您对于此 IdP 有其他客户端 ID（也称为受众），稍后可以在提供商详细信息页面上添加它们。

   > **Note**
   >
   > 如果 IdP JWT 令牌包含 `azp` 声明，请输入此值作为“受众”值。
   > 如果您的 OIDC 身份提供商在令牌中同时设置 `aud` 和 `azp` 声明，则 Amazon STS 将使用 `azp` 声明中的值作为 `aud` 声明。

7. （可选）对于 Add tags (添加标签)，您可以添加键值对来帮助识别和组织您的 IdP。您还可以使用标签来控制对 Amazon 资源的访问。要了解有关标记 IAM OIDC 身份提供程序的更多信息，请参阅 标记 OpenID Connect（OIDC）身份提供者。选择 Add tag（添加标签）。为每个标签键值对输入值。
8. 验证您提供的信息。完成此操作后，选择 Add provider（添加提供商）。IAM 将尝试检索并使用 OIDC IdP 服务器证书的顶级中间 CA 指纹来创建 IAM OIDC 身份提供者。

   > **Note**
   >
   > OIDC 身份提供者的证书链必须以域或颁发者 URL 开始，然后是中间证书，最后是根证书。如果证书链顺序不同或者包含重复证书或其他证书，则您会收到签名不匹配错

误，并且 STS 无法验证 JSON Web 令牌 (JWT)。更正服务器返回的链中证书的顺序以解决错误。有关证书链标准的更多信息，请参阅 RFC Series 网站上的 RFC 5246 中的 certificate_list。

9. 将 IAM 角色分配至身份提供程序，以向身份提供程序管理的外部用户身份授予访问账户中的 Amazon 资源的权限。要了解有关为联合身份创建角色的更多信息，请参阅 为第三方身份提供者创建角色

> **Note**
>
> 角色信任策略中使用的 OIDC IdP 必须与信任它的角色位于同一账户中。

为 IAM OIDC 身份提供程序添加或删除指纹（控制台）

> **Note**
>
> Amazon 使用我们的受信任根证书颁发机构（CA）库来保护与 OIDC 身份提供者（IdP）之间的通信，从而验证JSON Web 密钥集（JWKS）端点的 TLS 证书。如果您的 OIDC IdP 依赖的证书不是由其中某个受信任的 CA 签名，则仅在此时我们会使用 IdP 配置中设置的指纹来保护通信。如果我们无法检索 TLS 证书或需要 TLS v1.3，则 Amazon 将回退到指纹验证。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。然后选择要更新的 IAM 身份提供程序的名称。
3. 选择端点验证选项卡，然后在指纹部分中选择管理。要输入新的指纹值，请选择 Add thumbprint（添加指纹）。要删除指纹，请选择您要删除的指纹旁边的 Remove（删除）。

> **Note**
>
> IAM OIDC 身份提供程序必须具有至少 1 个、最多 5 个指纹。

完成后，选择 Save changes（保存更改）。

为 IAM OIDC 身份提供程序添加受众（控制台）

1. 在导航窗格中，选择 Identity providers（身份提供程序），然后选择要更新的 IAM 身份提供程序的名称。
2. 在 Audiences（受众）部分，选择 Actions（操作），然后选择 Add audience（添加受众）。
3. 键入您向 IdP 注册的、在 Step 1 中接收的并且将向 Amazon 发出请求的应用程序的客户端 ID。然后选择 Add audiences（添加受众）。

> **Note**
>
> IAM OIDC 身份提供程序必须具有至少 1 个且最多 100 个受众。

为 IAM OIDC 身份提供程序删除受众（控制台）

1. 在导航窗格中，选择 Identity providers（身份提供程序），然后选择要更新的 IAM 身份提供程序的名称。
2. 在 Audiences（受众）部分，选择要删除的受众旁边的单选按钮，然后选择 Actions（操作）。
3. 选择 Remove audience（删除受众）。此时会打开一个新窗口。
4. 如果删除受众，受众的联合身份将无法代入与受众关联的角色。在窗口中，阅读警告并通过在字段中键入 `remove` 一词以确认删除受众。
5. 选择 Remove（删除）以删除受众。

删除 IAM OIDC 身份提供程序（控制台）

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。
3. 选中要删除的 IAM 身份提供者旁边的复选框。此时会打开一个新窗口。
4. 通过在字段中键入 `delete` 一词以确认您要删除此提供商。然后选择 Delete(删除)。

创建和管理 IAM OIDC 身份提供程序 (Amazon CLI)

可以使用以下 Amazon CLI 命令来创建和管理 IAM OIDC 身份提供程序。

创建 IAM OIDC 身份提供程序 (Amazon CLI)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请运行以下命令：
   - aws iam list-open-id-connect-providers
2. 要创建新的 IAM OIDC 身份提供程序，请运行以下命令：
   - aws iam create-open-id-connect-provider

更新现有 IAM OIDC 身份提供程序的服务器证书指纹列表 (Amazon CLI)

- 要更新 IAM OIDC 身份提供程序的服务器证书指纹列表，请运行以下命令：
  - aws iam update-open-id-connect-provider-thumbprint

要标记现有 IAM OIDC 身份提供程序 (Amazon CLI)

- 要标记现有 IAM OIDC 身份提供程序，请运行以下命令：
  - aws iam tag-open-id-connect-provider

要列出现有 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要列出现有 IAM OIDC 身份提供程序的标签，请运行以下命令：
  - aws iam list-open-id-connect-provider-tags

要删除 IAM OIDC 身份提供程序 (Amazon CLI) 的标签

- 要删除现有 IAM OIDC 身份提供程序的标签，请运行以下命令：
  - aws iam untag-open-id-connect-provider

在现有 IAM OIDC 提供商中添加或删除客户端 ID (Amazon CLI)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请运行以下命令：
   - aws iam list-open-id-connect-providers
2. （可选）要获取有关 IAM OIDC 身份提供程序的详细信息，请运行以下命令：

- aws iam get-open-id-connect-provider

3. 要向现有 IAM OIDC 身份提供程序中添加新的客户端 ID，请运行以下命令：

   - aws iam add-client-id-to-open-id-connect-provider

4. 要从现有 IAM OIDC 身份提供程序中删除客户端，请运行以下命令：

   - aws iam remove-client-id-from-open-id-connect-provider

删除 IAM OIDC 身份提供程序 (Amazon CLI)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请运行以下命令：

   - aws iam list-open-id-connect-providers

2. （可选）要获取有关 IAM OIDC 身份提供程序的详细信息，请运行以下命令：

   - aws iam get-open-id-connect-provider

3. 要删除 IAM OIDC 身份提供程序，请运行以下命令：

   - aws iam delete-open-id-connect-provider

创建和管理 OIDC 身份提供程序 (Amazon API)

可以使用以下 IAM API 命令来创建和管理 OIDC 提供商。

创建 IAM OIDC 身份提供程序 (Amazon API)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请调用以下操作：

   - ListOpenIDConnectProviders

2. 要创建新的 IAM OIDC 身份提供程序，请调用以下操作：

   - CreateOpenIDConnectProvider

更新现有 IAM OIDC 身份提供程序的服务器证书指纹列表 (Amazon API)

- 要更新 IAM OIDC 身份提供程序的服务器证书指纹列表，请调用以下操作：

  - UpdateOpenIDConnectProviderThumbprint

要标记现有 IAM OIDC 身份提供程序 (Amazon API)

- 要标记现有 IAM OIDC 身份提供程序，请调用以下操作：
  - TagOpenIDConnectProvider

要列出现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

- 要列出现有 IAM OIDC 身份提供程序的标签，请调用以下操作：
  - ListOpenIDConnectProviderTags

要删除现有 IAM OIDC 身份提供程序 (Amazon API) 的标签

- 要删除现有 IAM OIDC 身份提供程序的标签，请调用以下操作：
  - UntagOpenIDConnectProvider

在现有 IAM OIDC 提供商中添加或删除客户端 ID (Amazon API)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请调用以下操作：
   - ListOpenIDConnectProviders
2. （可选）要获取有关 IAM OIDC 身份提供程序的详细信息，请调用以下操作：
   - GetOpenIDConnectProvider
3. 要向现有 IAM OIDC 身份提供程序中添加新的客户端 ID，请调用以下操作：
   - AddClientIDToOpenIDConnectProvider
4. 要从现有 IAM OIDC 身份提供程序中删除客户端 ID，请调用以下操作：
   - RemoveClientIDFromOpenIDConnectProvider

删除 IAM OIDC 身份提供程序 (Amazon API)

1. （可选）要获取您 Amazon 账户中所有 IAM OIDC 身份提供程序的列表，请调用以下操作：
   - ListOpenIDConnectProviders
2. （可选）要获取有关 IAM OIDC 身份提供程序的详细信息，请调用以下操作：

- GetOpenIDConnectProvider

3. 要删除 IAM OIDC 身份提供程序，请调用以下操作：

- DeleteOpenIDConnectProvider

获取 OpenID Connect 身份提供者的指纹

在 IAM 中创建 OpenID Connect（OIDC）身份提供商时，IAM 需要对外部身份提供者（IdP）使用的证书进行签名的顶级中间证书颁发机构（CA）的指纹。该指纹是用于为 OIDC 兼容 IdP 颁发证书的 CA 证书的签名。在创建 IAM OIDC 身份提供程序时，您信任该 IdP 验证的身份有权访问您的 Amazon Web Services 账户。通过使用 CA 的证书指纹，您信任该 CA 颁发的任何证书，并且其 DNS 名称与注册的名称相同。这样，在续订 IdP 的签名证书时，无需在每个账户中更新信任关系。

> **Important**
>
> 在大多数情况下，联合身份验证服务器使用两个不同的证书：
>
> - 第一个证书将在 Amazon 与您的 IdP 之间建立 HTTPS 连接。这应由已知公有根 CA（如 Amazon Certificate Manager）发布。这使客户端能够检查证书的可靠性和状态。
> - 第二个证书将用于加密令牌，且应由私有或公有根 CA 签署。

您可以使用 Amazon Command Line Interface、Tools for Windows PowerShell 或 IAM API 创建一个 IAM OIDC 身份提供程序。使用这些方法时，您可以选择手动提供指纹。如果您选择不包含指纹，IAM 将检索 OIDC IdP 服务器证书的顶级中间 CA 指纹。如果您选择包含指纹，则必须手动获取指纹并将它提供给 Amazon。

当您使用 IAM 控制台创建 OIDC 身份提供者时，IAM 会尝试为您检索 OIDC IdP 服务器证书的顶级中间 CA 指纹。

此外，建议您手动获取 OIDC IdP 的指纹，并验证 IAM 是否检索了正确的指纹。有关获取证书指纹的更多信息，请参阅以下部分：

> **Note**
>
> Amazon 使用我们的受信任根证书颁发机构（CA）库来保护与 OIDC 身份提供者（IdP）之间的通信，从而验证JSON Web 密钥集（JWKS）端点的 TLS 证书。如果您的 OIDC IdP 依赖的

证书不是由其中某个受信任的 CA 签名，则仅在此时我们会使用 IdP 配置中设置的指纹来保护通信。如果我们无法检索 TLS 证书或需要 TLS v1.3，则 Amazon 将回退到指纹验证。

Important

如果您的 OIDC 身份提供者使用不同的主机作为发现端点（发布者 URL）和 JWKS 端点（`jwks_uri`），则必须在 OIDC 提供者的指纹列表中包含两个端点的指纹。回退到指纹验证时，Amazon 会验证两个端点的指纹。如果配置的列表中缺少任一指纹，则身份验证将失败。大多数 OIDC 提供者将两个端点托管在同一个域上，因此一个指纹可以覆盖这两个端点。只有当您的提供者为两个端点使用不同的主机或证书时，您才需要添加单独的指纹。

获取证书指纹

您可以使用 Web 浏览器和 OpenSSL 命令行工具获取 OIDC 提供者的证书指纹。但是，您无需手动获取证书指纹即可创建 IAM OIDC 身份提供者。您可以使用以下过程获取 OIDC 提供者。

获取 OIDC IdP 的指纹

1. 您需要先获取 OpenSSL 命令行工具，然后才能获取 OIDC IdP 的指纹。您可使用此工具下载 OIDC IdP 的证书链并生成证书链中最终证书的指纹。如果需要安装和配置 OpenSSL，请遵循安装 OpenSSL 和配置 OpenSSL 中的说明。
2. 从 OIDC IdP 的 URL 开始（例如，`https://server.example.com`），然后添加 `/.well-known/openid-configuration` 以构成该 IdP 的配置文档的 URL，如下所示：

   `https://server.example.com/.well-known/openid-configuration`

   在 Web 浏览器中打开此 URL，将 *`server.example.com`* 替换为 IdP 的服务器名称。
3. 在显示的文档中，使用 Web 浏览器 Find（查找）功能来定位文本 `"jwks_uri"`。`"jwks_uri"` 文本后面会跟有一个冒号 (:)，然后是一个 URL。复制 URL 的完全限定域名。不包括 `https://` 或在顶级域后的任何路径。

```
{
 "issuer": "https://accounts.example.com",
 "authorization_endpoint": "https://accounts.example.com/o/oauth2/v2/auth",
 "device_authorization_endpoint": "https://oauth2.exampleapis.com/device/code",
 "token_endpoint": "https://oauth2.exampleapis.com/token",
 "userinfo_endpoint": "https://openidconnect.exampleapis.com/v1/userinfo",
```

```
 "revocation_endpoint": "https://oauth2.exampleapis.com/revoke",
 "jwks_uri": "https://www.exampleapis.com/oauth2/v3/certs",
...
```

> **Note**
>
> 如果您的发现和 JWKS 端点使用不同的主机，请记下您的发布者 URL 中的完全限定域名（您在 Step 2 中使用的 URL，不带 `/.well-known/openid-configuration` 路径）。如果此域与您在此步骤中复制的 `jwks_uri` 域不同，请在创建 OIDC 提供者时对两个域执行 Step 4 至 Step 6 并包含两个指纹。

4. 使用 OpenSSL 命令行工具可运行以下命令。将 *keys.example.com* 替换为您在Step 3 中获取的域名。

```
openssl s_client -servername keys.example.com -showcerts -
connect keys.example.com:443
```

5. 在命令窗口中向上滚动，直至看到类似于以下示例的证书。如果您查看多个证书，请找到显示的最后一个证书（在命令输出底部）。这包含证书颁发机构链中的顶级中间的 CA 的证书。

```
-----BEGIN CERTIFICATE-----
 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
 21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
 rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
 nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
 NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
 -----END CERTIFICATE-----
```

复制证书（包括 `-----BEGIN CERTIFICATE-----` 和 `-----END CERTIFICATE-----` 行）并将其粘贴到文本文件中。然后使用文件名 **`certificate.crt`** 保存该文件。

> **Note**
>
> OIDC 身份提供者的证书链必须以域或颁发者 URL 开始，包含任何中间证书（如果有），然后以根证书结尾。如果证书链顺序不同或者包含重复证书或其他证书，您将收到签名不匹配错误，并且 STS 无法验证 JSON Web 令牌（JWT）。更正服务器返回的链中证书的顺序以解决错误。有关证书链标准的更多信息，请参阅 RFC Series 网站上的 RFC 5246 中的 certificate_list。

6. 使用 OpenSSL 命令行工具可运行以下命令。

```
openssl x509 -in certificate.crt -fingerprint -sha1 -noout
```

您的命令窗口将显示类似于以下示例的证书指纹：

```
SHA1 Fingerprint=99:0F:41:93:97:2F:2B:EC:F1:2D:DE:DA:52:37:F9:C9:52:F2:0D:9E
```

请从此字符串中去掉冒号 (:) 字符以生成最终指纹，如：

```
990F4193972F2BECF12DDEDA5237F9C952F20D9E
```

> **Note**
>
> 如果您的发现端点使用的主机与 JWKS 端点不同，请使用发现端点域（您的发布者 URL 域）重复 Step 4 至 Step 6。使用 CreateOpenIDConnectProvider 或 UpdateOpenIDConnectProviderThumbprint 创建或更新 OIDC 身份提供者时，包括两个指纹。

7. 如果要使用 Amazon CLI、Tools for Windows PowerShell 或 IAM API 创建 IAM OIDC 身份提供者，则提供指纹是可选的。如果您选择在创建过程中不包含指纹，IAM 将检索 OIDC IdP 服务器证书的顶级中间 CA 指纹。创建 IAM OIDC 身份提供者后，您可以将此指纹与 IAM 检索到的指纹进行比较。

   如果您在 IAM 控制台中创建 IAM OIDC 身份提供者，则控制台会尝试为您检索 OIDC IdP 服务器证书的顶级中间 CA 指纹。您可以将此指纹与 IAM 检索到的指纹进行比较。创建 IAM OIDC 身份提供者后，您可以在 OIDC 提供者摘要控制台页面的端点验证选项卡中查看IAM OIDC 身份提供者的指纹。

> **Important**
>
> 如果您获得的指纹与您在 IAM OIDC 身份提供者指纹详细信息中看到的指纹不匹配，则不应使用 OIDC 提供者。相反，您应该删除已创建的 OIDC 提供者，然后在一段时间后再次尝试创建 OIDC 提供者。在使用提供者之前，请验证指纹是否匹配。如果第二次尝试后指纹仍然不匹配，请使用 IAM 论坛联系 Amazon。

安装 OpenSSL

如果您没有安装 OpenSSL，请按照本部分中的说明执行操作。

在 Linux 或 Unix 系统上安装 OpenSSL

1. 转到 OpenSSL：源、Tarball (https://openssl.org/source/)。
2. 下载最新源并构建包。

如要在 Windows 系统上安装 OpenSSL

1. 转到 OpenSSL：二进制分布 (https://wiki.openssl.org/index.php/Binaries)，了解您可以从中安装 Windows 版本的站点列表。
2. 按照所选站点上的说明开始安装。
3. 如果系统要求您安装 Microsoft Visual C++ 2008 Redistributables 并且该程序尚未安装在您的系统上，请选择适合您环境的下载链接。按照 Microsoft Visual C++ 2008 Redistributable 安装向导中的说明操作。

> **Note**
>
> 如果您不确定您的系统上是否已安装 Microsoft Visual C++ 2008 Redistributables，则可以尝试先安装 OpenSSL。如果尚未安装 Microsoft Visual C++ 2008 Redistributables，OpenSSL 安装程序将显示提示。请确保安装与您安装的 OpenSSL 版本匹配的体系架构（32 位或 64 位）。

4. 在安装 Microsoft Visual C++ 2008 Redistributables 后，为您的环境选择适当的 OpenSSL 二进制版本，然后在本地保存该文件。启动 OpenSSL 设置向导。
5. 按照 OpenSSL 设置向导中的说明进行操作。

配置 OpenSSL

在使用 OpenSSL 命令之前，您必须配置操作系统，使其具有有关 OpenSSL 安装位置的信息。

要在 Linux 或 Unix 上配置 OpenSSL

1. 在命令行中，将 OpenSSL_HOME 变量设置为 OpenSSL 安装的位置：

```
$ export OpenSSL_HOME=path_to_your_OpenSSL_installation
```

2. 设置包含 OpenSSL 安装的路径：

```
$ export PATH=$PATH:$OpenSSL_HOME/bin
```

> **Note**
>
> 通过使用 export 命令行对环境变量所做的任何更改只对当前的会话有效。通过在 shell 配置文件中设置环境变量，可对环境变量进行持续更改。有关更多信息，请参阅您的操作系统文档。

要在 Windows 上配置 OpenSSL

1. 打开 Command Prompt（命令提示符窗口）。
2. 将 OpenSSL_HOME 变量设置为 OpenSSL 安装的位置：

```
C:\> set OpenSSL_HOME=path_to_your_OpenSSL_installation
```

3. 将 OpenSSL_CONF 变量设置为 OpenSSL 安装中配置文件的位置：

```
C:\> set OpenSSL_CONF=path_to_your_OpenSSL_installation\bin\openssl.cfg
```

4. 设置包含 OpenSSL 安装的路径：

```
C:\> set Path=%Path%;%OpenSSL_HOME%\bin
```

> **Note**
>
> 通过 Command Prompt（命令提示符）对 Windows 环境变量所做的任何更改只对当前的命令行会话有效。您可以通过将环境变量设置为系统属性来对其进行持久性更改。确切的流程取决于您使用的 Windows 版本。（例如，在 Windows 7 中，打开 Control Panel（控制面板）、System and Security（系统和安全性）、System（系统）。然后选择 Advanced system settings（高级系统设置）、Advanced（高级）选项卡，Environment Variables（环境变量）。）有关更多信息，请参阅 Windows 文档。

适用于共享 OIDC 提供者的身份提供者控制

对于公认的共享 OpenID Connect（OIDC）身份提供者（IdP），IAM 要求对角色信任策略中的特定声明进行明确评估。这些必需的声明名为身份提供者控制，由 IAM 在角色创建和信任策略更新期间进行评估。如果角色信任策略未对共享 OIDC IdP 所需的控制进行评估，则角色创建或更新将失败。这样可以确保，只有预期组织的授权身份才能代入角色和访问 Amazon 资源。这种安全控制在多个 Amazon 客户共享 OIDC 提供者时至关重要。

IAM 不会针对现有 OIDC 角色信任策略对身份提供者控制进行评估。对于针对现有 OIDC 角色的角色信任策略所进行的任何修改，IAM 将要求在角色信任策略中包含身份提供者控制。

OIDC 提供者类型

IAM 将 OIDC 身份提供者分为两种不同的类型：私有和共享。私有 OIDC IdP 可以由单个组织拥有和管理，也可以是 SaaS 提供商的租户，其 OIDC 颁发者 URL 是该组织特有的唯一标识符。相比之下，多个组织使用共享的 OIDC IdP，其中使用该共享身份提供者的所有组织的 OIDC 颁发者 URL 可能相同。

下表概述了私有和共享 OIDC 提供者之间的主要区别：

特征	私有 OIDC 提供者	共有 OIDC 提供者
Issuer	归组织独有	在多个组织之间共享
租赁信息	通过唯一的颁发者进行沟通	通过 JWT 中的声明进行沟通
信任策略要求	无需进行特定的声明评估	需要对特定声明进行评估

具有身份提供者控制的共享 OIDC 身份提供者

当您在 IAM 中创建或修改 OIDC 提供者时，系统会自动识别和评估公认的共享 OIDC 提供者所需的声明。如果未在角色信任策略中配置身份提供者控制，则角色创建或更新将失败，并出现 MalformedPolicyDocument 错误。

下表列出了在角色信任策略中需要身份提供者控制的共享 OIDC 提供者以及可帮助您配置身份提供者控制的其他信息。

OIDC IdP	OIDC URL	租赁声明	所需声明
Amazon Cognito	`cognito-identity.cn-north-1.amazonaws.com.cn`	aud	`cognito-identity.cn-north-1.amazonaws.com.cn:aud`
Azure Sentinel	`https://sts.windows.net/33e01921-4d64-4f8c-a055-5bdaffd5e33d`	`sts:RoleSessionName`	`sts:RoleSessionName`
Buildkite	`https://agent.buildkite.com`	sub	`agent.buildkite.com:sub`
Codefresh SaaS	`https://oidc.codefresh.io`	sub	`oidc.codefresh.io:sub`
DVC Studio	`https://studio.datachain.ai/api`	sub	`studio.datachain.ai/api:sub`
GitHub 操作	`https://token.acti`	sub or job_workflow_ref	`token.actions.gith`

OIDC IdP	OIDC URL	租赁声明	所需声明
	ons.githubusercontent.com		ubusercontent.com:sub 或者 token.actions.githubusercontent.com:job_workflow_ref
GitHub 审核日志流式传输	https://oidc-configuration.audit-log.githubusercontent.com	sub	oidc-configuration.audit-log.githubusercontent.com:sub
GitHub vstoken	https://vstoken.actions.githubusercontent.com	sub	vstoken.actions.githubusercontent.com:sub
GitLab	https://gitlab.com	sub	gitlab.com:sub

OIDC IdP	OIDC URL	租赁声明	所需声明
IBM Turbonomic SaaS*	• https://rh-oidc.s3.us-east-1.amazonaws.com/22ejnvnnturfmt6km08iidd0nt4hekbn7 • https://rh-oidc.s3.us-east-1.amazonaws.com/23e3sd27sju1hoou6ohfs68vbno607tr • https://rh-oidc.s3.us-east-1.amazonaws.com/23ne21h005qjl3n33d8dui5dlrmv2tmg • https://rh-oidc.s3.us-east-1.amazonaws.com/24jrf12m5dj7ljlfb4ta2frhrcoadm26	sub	• rh-oidc.s3.us-east-1.amazonaws.com/22ejnvnnturfmt6km08iidd0nt4hekbn7:sub • rh-oidc.s3.us-east-1.amazonaws.com/23e3sd27sju1hoou6ohfs68vbno607tr:sub • rh-oidc.s3.us-east-1.amazonaws.com/23ne21h005qjl3n33d8dui5dlrmv2tmg:sub • rh-oidc.s3.us-east-1.amazonaws.com/24jrf12m5dj7ljlfb4ta2frhrcoadm26:sub

OIDC IdP	OIDC URL	租赁声明	所需声明
	• https://oidc.op1.openshiftapps.com/2f785sojlpb85i7402pk3qogugim5nfb • https://oidc.op1.openshiftapps.com/2c51blsaqa9gkjt0o9rt11mle8mmropu		• oidc.op1.openshiftapps.com/2f785sojlpb85i7402pk3qogugim5nfb:sub • oidc.op1.openshiftapps.com/2c51blsaqa9gkjt0o9rt11mle8mmropu:sub
Pulumi Cloud	https://api.pulumi.com/oidc	aud	api.pulumi.com/oidc:aud
sandboxes.cloud	https://sandboxes.cloud	aud	sandboxes.cloud:aud
Scalr	https://scalr.io	sub	scalr.io:sub
Shisho Cloud	https://tokens.cloud.shisho.dev	sub	tokens.cloud.shisho.dev:sub
Terraform Cloud	https://app.terraform.io	sub	app.terraform.io:sub

OIDC IdP	OIDC URL	租赁声明	所需声明
Upbound	`https://proidc.upbound.io`	sub	`proidc.upbound.io:sub`
Vercel 全局端点	`https://oidc.vercel.com`	aud	`oidc.vercel.com:aud`

* IBM Turbonomic 定期使用该平台的新版本更新其 OIDC 颁发者 URL。我们将根据需要在范围内增加其他 Turbonomic OIDC 颁发者作为共享提供商。

对于 IAM 确定为共享的任何新的 OIDC IdP，将以类似的方式记录和执行角色信任策略所需的身份提供者控制。

其他资源

其他资源

- 有关如何为 OIDC 联合身份验证创建 IAM 角色的更多信息，请参阅 创建用于 OpenID Connect 联合身份验证（控制台）的角色。
- 有关可用于声明的 IAM 条件键的列表，请参阅 Amazon OIDC 联合身份验证的可用键。

SAML 2.0 联合身份验证

Amazon 使用 SAML 2.0（安全断言标记语言 2.0）支持身份联合验证，SAML 2.0 是许多身份提供者 (IdP) 使用的一种开放标准。此功能可实现联合单点登录 (SSO)，因此用户可以登录 Amazon Web Services 管理控制台 或调用 Amazon API 操作，而不必为企业中的每个人都创建一个 IAM 用户。通过使用 SAML，您可以使用 Amazon 简化配置联合身份验证流程，因为您可以使用 IdP 的服务而不是编写自定义身份代理代码。

> **Note**
>
> IAM SAML 身份联合验证支持来自基于 SAML 的联合身份提供者（IdP）的加密 SAML 响应。IAM Identity Center 和 Amazon Cognito 不支持来自 IAM SAML 身份提供商的加密 SAML 断言。
>
> 您可以将对加密 SAML 断言的支持间接添加到具有 Amazon Cognito 用户池的 Amazon Cognito 身份池联合身份验证。用户池具有独立于 IAM SAML 联合身份验证的 SAML 联合身份

验证，并且支持 SAML 签名和加密。尽管此功能没有直接扩展到身份池，但用户池可以是身份池的 IDP。要对身份池使用 SAML 加密，请将具有加密的 SAML 提供商添加到一个用户池，其是身份池的 IdP。
SAML 提供商必须能够使用用户池提供的密钥对 SAML 断言进行加密。用户池不接受使用 IAM 提供的证书进行加密的断言。

IAM 联合身份验证支持这些使用案例：

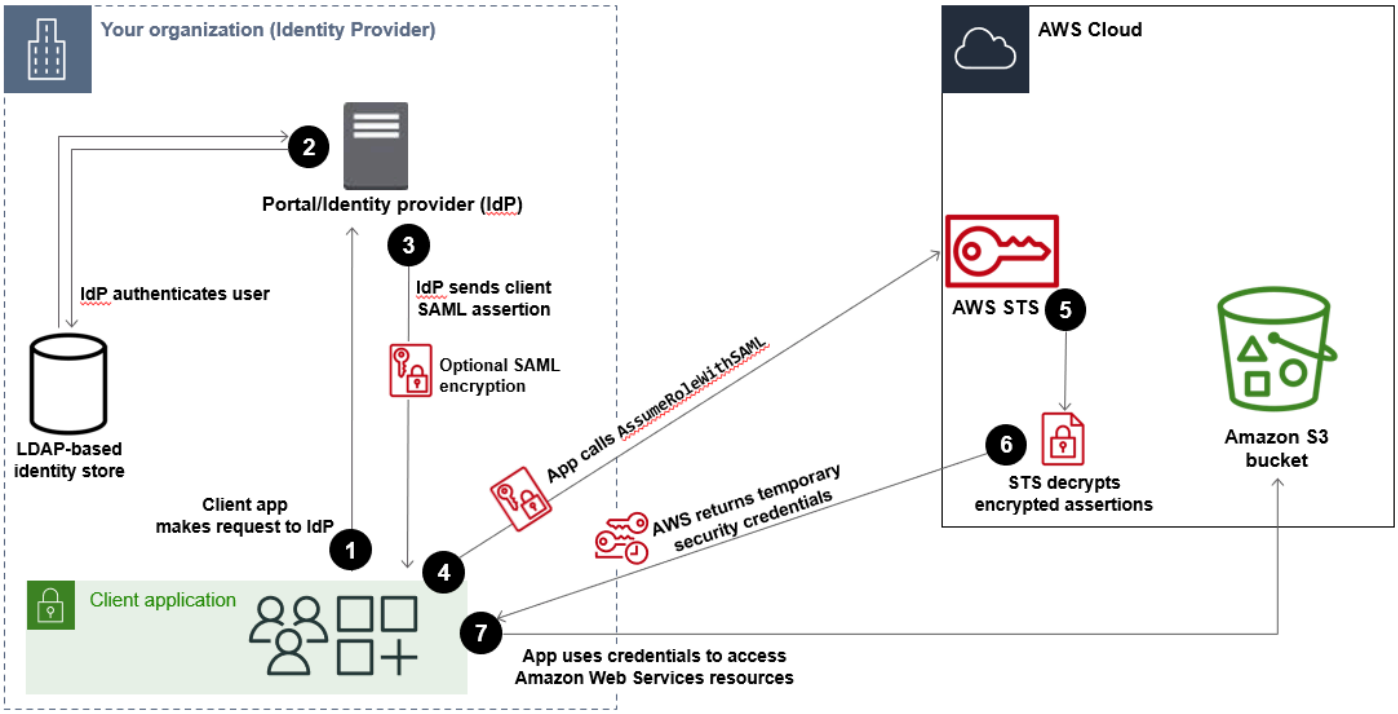
- 允许组织中的用户或应用程序调用 Amazon API 操作的联合访问权限。下面的部分将讨论此用例。您可以使用组织内生成的 SAML 断言 (身份验证响应的一部分) 获得临时安全凭证。此方案类似于 IAM 支持的其他联合身份验证方案，如 请求临时安全凭证 和 OIDC 联合身份验证 中介绍的方案。但是，企业中基于 SAML 2.0 的 IdP 可以在运行时处理很多细节功能，以用于执行身份验证和授权检查。
- 从组织向 Amazon Web Services 管理控制台进行基于 Web 的单点登录 (SSO)。用户可以登录您企业中由与 SAML 2.0 兼容的 IdP 托管的门户，选择转向 Amazon，并将其重新导向到控制台，而无需提供其他登录信息。您可以使用第三方 SAML IdP 建立对控制台的 SSO 访问，或者可以创建自定义 IdP 来支持外部用户的控制台访问。有关构建自定义 IdP 的更多信息，请参阅使自定义身份凭证代理程序能够访问 Amazon 控制台。

主题

- 使用基于 SAML 的联合身份验证来对 Amazon 进行 API 访问
- 配置基于 SAML 2.0 的联合身份验证的概述
- 用于允许对 Amazon 资源进行 SAML 联合访问的角色的概述
- 唯一标识基于 SAML 的联合身份验证中的用户
- 在 IAM 中创建 SAML 身份提供者
- 配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述
- 将第三方 SAML 解决方案提供者与 Amazon 集成
- 为身份验证响应配置 SAML 断言。
- 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台
- 在您的浏览器中查看 SAML 响应

使用基于 SAML 的联合身份验证来对 Amazon 进行 API 访问

假设您想要为员工提供一种将数据从他们的计算机中复制到备份文件夹的方法。您可以构建一个可在用户的计算机上运行的应用程序。在后端，该应用程序可在 Amazon S3 存储桶中读写对象。用户没有直接访问 Amazon 的权限。而应使用以下过程：



1. 您组织中的用户使用客户端应用程序来请求您组织的 IdP 进行身份验证。
2. IdP 根据组织的身份存储对用户进行身份验证。
3. IdP 构建一个具有用户相关信息的 SAML 断言，并将此断言发送到客户端应用程序。当您为 IAM SAML IdP 启用 SAML 加密时，此断言将由外部 IdP 加密。
4. 客户端应用程序调用 Amazon STS `AssumeRoleWithSAML` API，并传递 SAML 提供商的 ARN、要代入的角色的 ARN 以及来自 IdP 的 SAML 断言。如果启用了加密，则通过客户端应用程序传递的断言在传输过程中保持加密状态。
5. （可选）Amazon STS 使用您从外部 IdP 上传的私有密钥来解密加密的 SAML 断言。
6. API 对客户端应用程序的响应包括临时安全凭证。
7. 客户端应用程序使用临时安全凭证来调用 Amazon S3 API 操作。

配置基于 SAML 2.0 的联合身份验证的概述

在使用前面方案和图表中所述的基于 SAML 2.0 的联合身份验证之前，您必须先配置组织的 IdP 和您的 Amazon Web Services 账户，使之相互信任。以下步骤介绍了用于配置此信任的一般过程。组织内部必须有支持 SAML 2.0 的 IdP，例如 Microsoft Active Directory 联合身份验证服务 (AD FS，Windows Server 的一部分)、Shibboleth 或其他兼容的 SAML 2.0 提供商。

> **Note**
>
> 为了提高联合身份验证弹性，我们建议您将 IdP 和Amazon联合身份验证配置为支持多个 SAML 登录端点。有关详细信息，请参阅 Amazon 安全博客文章如何使用区域性 SAML 端点进行失效转移。

配置组织的 IdP 和 Amazon 以使之相互信任

1. 将 Amazon 注册为您组织的 IdP 的服务提供商（SP）。通过 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 使用 SAML 元数据文档

   有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。

   您可选择使用 `https://signin.aws.amazon.com/static/saml-metadata.xml` 的 SAML 元数据文档。

2. 通过使用您企业的 IdP，生成一个同等 SAML 元数据 XML 文件，该文件可将您的 IdP 描述为 Amazon 中的 IAM 身份提供者。它必须包括发布者名称、创建日期、过期日期以及 Amazon 可用来验证来自您组织的身份验证响应（断言）的密钥。

   如果允许从外部 IdP 发送加密的 SAML 断言，则必须使用组织的 IdP 生成私有密钥文件，并以 .pem 文件格式将此文件上传到 IAM SAML 配置。Amazon STS 需要此私有密钥文件来解密与上传到 IdP 的公有密钥对应的 SAML 响应。

   > **Note**
   >
   > 如 SAML V2.0 元数据互操作性配置文件 1.0 版所定义，IAM 既不会评估 SAML 元数据文档的 X.509 证书，也不会在该证书过期时采取任何行动。如果您担心 X.509 证书过期，建议您监控证书到期日期，并根据贵组织的治理和安全策略来轮换证书。

3. 在 IAM 控制台中，创建一个 SAML 身份提供商。在此过程中，您将上传 SAML 元数据文档和私有解密密钥，这些文档和私有解密密钥是由贵组织中的 IdP 在 Step 2 中生成的。有关更多信息，请参阅 在 IAM 中创建 SAML 身份提供者。
4. 在 IAM 中创建一个或多个 IAM 角色。在角色的信任策略中，您可将 SAML 提供商设置为可在您的组织与 Amazon 之间建立信任关系的主体。该角色的权限策略确定了允许您组织的用户在 Amazon 中执行的操作。有关更多信息，请参阅 为第三方身份提供者创建角色。

   > **Note**
   >
   > 角色信任策略中使用的 SAML IdP 必须与角色位于同一账户中。

5. 在您企业的 IdP 中，定义可将您企业中的用户或组映射到 IAM 角色的断言。请注意，您的组织中不同的用户和组可能映射到不同的 IAM 角色。执行映射的确切步骤取决于您使用的 IdP。在用户 Amazon S3 文件夹中的较早场景中，则所有用户都可能映射到提供 Amazon S3 权限的同一角色。有关更多信息，请参阅 为身份验证响应配置 SAML 断言。。

   如果您的 IdP 支持对 Amazon 控制台的 SSO，则可配置控制台会话的最大持续时间。有关更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。
6. 在您正在创建的应用程序中，您可以调用 Amazon Security Token Service `AssumeRoleWithSAML` API，将其传递给您在Step 3 中创建的 SAML 提供商的 ARN、您在Step 4 中创建的要代入的角色的 ARN 以及从您的 IdP 处获取的有关当前用户的 SAML 断言。Amazon 确保代入角色的请求来自 SAML 提供商所引用的 IdP。

   有关更多信息，请参阅https://docs.amazonaws.cn/STS/latest/APIReference/API_AssumeRoleWithSAML.html API 参考中的 Amazon Security Token ServiceAssumeRoleWithSAML。
7. 如果请求成功，API 会返回一组临时安全凭证，您的应用程序即可用其向 Amazon 发出已签名的请求。您的应用程序具有有关当前用户的信息并可访问 Amazon S3 中用户特定的文件夹，如上一方案中所述。

用于允许对 Amazon 资源进行 SAML 联合访问的角色的概述

您在 IAM 中创建的角色将确定您组织中的 SAML 联合主体在 Amazon 中允许执行的操作。当您为角色创建信任策略时，您可以将先前创建的 SAML 提供商指定为 `Principal`。此外，您还可以使用 `Condition` 设置信任策略的范围，以便仅允许与特定 SAML 属性匹配的用户访问角色。例如，您可以指定仅允许 SAML 从属关系为 `staff` (在 https://openidp.feide.no 中断言) 的用户访问角色，如以下示例策略所示：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"Federated": "arn:aws:iam::111122223333:saml-provider/
ExampleOrgSSOProvider"},
    "Action": "sts:AssumeRoleWithSAML",
    "Condition": {
      "StringEquals": {
        "saml:aud": "https://us-east-1.signin.aws.amazon.com/saml",
        "saml:iss": "https://openidp.feide.no"
      },
      "ForAllValues:StringLike": {"saml:edupersonaffiliation": ["staff"]}
    }
  }]
}
```

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws-cn:iam::111122223333:saml-provider/
ExampleOrgSSOProvider"
            },
            "Action": "sts:AssumeRoleWithSAML",
            "Condition": {
                "StringEquals": {
                    "saml:aud": "https://ap-east-1.signin.amazonaws.cn/saml",
                    "saml:iss": "https://openidp.feide.no"
                },
                "ForAllValues:StringLike": {
                    "saml:edupersonaffiliation": [
                        "staff"
                    ]
                }
```

```
            }
        }
    ]
}
```

> **Note**
>
> 角色信任策略中使用的 SAML IdP 必须与角色位于同一账户中。

策略中的 `saml:aud` 上下文键指定登录控制台时浏览器显示的 URL。此登录端点 URL 必须与您的身份提供者的收件人属性相匹配。您可以添加特定区域内的登录 URL。Amazon 建议使用区域端点而不是全局端点，以提高联合身份验证的韧性。如果您只配置了一个端点，则如果该端点变得不可用（此情况发生可能性极小），您就无法通过联合身份验证登录 Amazon。有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。

以下示例显示了带有可选 `region-code` 的登录 URL 格式。

`https://`*region-code*`.signin.aws.amazon.com/saml`

如果需要 SAML 加密，则登录 URL 必须包含 Amazon 分配给您的 SAML 提供商的唯一标识符，您可以在身份提供者详细信息页面上找到该标识符。在以下示例中，登录 URL 包含要求在登录路径后附加 /acs/ 的 IdP 唯一标识符。

`https://`*region-code*`.signin.aws.amazon.com/saml/acs/`*IdP-ID*

对于该角色中的权限策略，您可以像任何角色一样指定权限。例如，如果允许您企业的用户管理 Amazon Elastic Compute Cloud 实例，您必须在权限策略中明确允许 Amazon EC2 操作，如 AmazonEC2FullAccess 托管策略中的操作。

有关您可以签入策略的 SAML 密钥的更多信息，请参阅基于 SAML 的 Amazon STS 联合身份验证的可用键。

唯一标识基于 SAML 的联合身份验证中的用户

在 IAM 中创建访问策略时，可根据用户的身份指定权限，这一点通常很有用。举例来说，对于已使用 SAML 联合的用户，应用程序可能希望使用如下的结构保留 Amazon S3 中的信息：

```
amzn-s3-demo-bucket/app1/user1
amzn-s3-demo-bucket/app1/user2
```

```
amzn-s3-demo-bucket/app1/user3
```

您可以通过 Amazon S3 控制台或 Amazon CLI 创建存储桶 (amzn-s3-demo-bucket) 和文件夹 (app1)，因为这些都是静态值。但是，用户特定文件夹（*user1*、*user2*、*user3* 等）必须在运行时使用代码创建，因为在用户首次通过联合身份验证之前，用来标识用户的值是未知的。

要编写在资源名称中引用特定于用户的详细信息的策略，必须在可以用于策略条件的 SAML 密钥中提供用户身份。以下密钥可用于基于 SAML 2.0 的联合身份验证，以便在 IAM 策略中使用。您可以使用以下键返回的值为资源 (如 Amazon S3 文件夹) 创建唯一的用户标识符。

- saml:namequalifier. 哈希值，基于 Issuer 响应值 (saml:iss)、包含 AWS 账户 ID 的字符串和 IAM 中 SAML 提供商的友好名称（ARN 的最后一部分）的串联。账户 ID 与 SAML 提供商的易记名称的串联可作为键 saml:doc 供 IAM 策略使用。账户 ID 与提供商名称必须使用“/”分隔，例如在“123456789012/provider_name”中。有关更多信息，请参阅saml:doc 上的 基于 SAML 的 Amazon STS 联合身份验证的可用键 键。

  NameQualifier 与 Subject 的组合可用于唯一识别 SAML 联合主体。以下伪代码显示如何计算此值。在此伪代码中，+ 表示串联，SHA1 代表使用 SHA-1 生成消息摘要的功能，Base64 代表生成哈希输出的 Base-64 编码版本的功能。

  ```
  Base64 ( SHA1 ( "https://example.com/saml" + "123456789012" + "/
  MySAMLIdP" ) )
  ```

  有关可用于基于 SAML 的联合身份验证的策略键的更多信息，请参阅基于 SAML 的 Amazon STS 联合身份验证的可用键。
- saml:sub（字符串）。这是该陈述的主题，其中包含唯一标识组织中某个用户的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。
- saml:sub_type（字符串）。此键可以是 persistent、transient 或在您的 SAML 断言中使用的 Format 和 Subject 元素的完整 NameID URI。persistent 的值表示在所有会话中用户的 saml:sub 值是相同的。如果值为 transient，则用户在每个会话中拥有不同的 saml:sub 值。有关 NameID 元素的 Format 属性的信息，请参阅为身份验证响应配置 SAML 断言。。

以下示例说明了一个权限策略，该策略使用上述密钥为 Amazon S3 中的用户特定文件夹授予权限。该策略假设 Amazon S3 对象使用同时包含 saml:namequalifier 和 saml:sub 的前缀进行标识。请注意，Condition 元素包括一个测试，用于确保 saml:sub_type 设置为 persistent。如果已设置为 transient，每个会话用户的 saml:sub 值可以不同，且不应使用值的组合来标识用户特定的文件夹。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws:s3:::amzn-s3-demo-bucket-org-data/backup/${saml:namequalifier}/
${saml:sub}",
      "arn:aws:s3:::amzn-s3-demo-bucket-org-data/backup/${saml:namequalifier}/
${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
}
```

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "s3:GetObject",
      "s3:PutObject",
      "s3:DeleteObject"
    ],
    "Resource": [
      "arn:aws-cn:s3:::amzn-s3-demo-bucket-org-data/backup/${saml:namequalifier}/
${saml:sub}",
      "arn:aws-cn:s3:::amzn-s3-demo-bucket-org-data/backup/${saml:namequalifier}/
${saml:sub}/*"
    ],
    "Condition": {"StringEquals": {"saml:sub_type": "persistent"}}
  }
```

```
}
```

有关将断言从 IdP 映射到策略的更多信息，请参阅为身份验证响应配置 SAML 断言。。

在 IAM 中创建 SAML 身份提供者

IAM SAML 2.0 身份提供程序是 IAM 中的一个实体，该实体描述支持 SAML 2.0 (安全断言标记语言 2.0) 标准的外部身份提供程序 (IdP) 服务。如果您希望在与 SAML 兼容的 IdP（例如 Shibboleth 或 Active Directory 联合身份验证服务）和 Amazon 之间建立信任，以便企业中的用户能够访问 Amazon 资源，则需要使用 IAM 身份提供者。IAM SAML 身份提供程序用作 IAM 信任策略中的主体。

有关此方案的更多信息，请参阅SAML 2.0 联合身份验证。

您可以在 Amazon Web Services 管理控制台 中或通过使用 Amazon CLI、Tools for Windows PowerShell 或 Amazon API 调用，创建和管理 IAM 身份提供程序。

创建 SAML 提供商后，必须创建一个或多个 IAM 角色。角色是 Amazon 中的一个实体，它没有自己的凭证（与用户一样）。但在此情况下，角色将动态分配给由 IdP 验证的 SAML 联合主体。该角色允许您的 IdP 请求临时安全凭证以便访问 Amazon。分配给该角色的策略决定了用户可在 Amazon 中执行的操作。要创建用于 SAML 联合的角色，请参阅为第三方身份提供者创建角色。

最后，在创建角色后，您可通过配置包含有关 Amazon 的信息的 IdP 以及希望 SAML 联合主体使用的角色来完成 SAML 信任。这称为在 IdP 和 Amazon 之间配置信赖方信任。要配置信赖方信任，请参阅配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述。

主题

- 先决条件
- 创建和管理 IAM SAML 身份提供商（控制台）
- 管理 SAML 加密密钥
- 创建和管理 IAM SAML 身份提供商 (Amazon CLI)
- 创建和管理 IAM SAML 身份提供商 (Amazon API)
- 后续步骤

先决条件

在创建 SAML 身份提供商之前，您必须从 IdP 处获得以下信息。

- 从 IdP 中获取 SAML 元数据文档。此文档包括发布者名称、过期信息以及可用来验证从 IdP 处收到的 SAML 身份验证响应（断言）的密钥。要生成元数据文档，请使用外部 IdP 提供的身份管理软件。

  > **Important**
  >
  > 此元数据文件包括颁发者名称、过期信息以及可用来验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。要删除 BOM，您可以使用 Notepad++ 等文本编辑工具以 UTF-8 格式对文件进行编码。
  > 作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。此外，X.509 证书也不能有任何重复的扩展名。您可以使用扩展程序，但扩展程序只能在证书中显示一次。如果 X.509 证书不符合任一条件，则 IdP 将创建失败，并返回“Unable to parse metadata”这一错误消息。
  > 如 SAML V2.0 元数据互操作性配置文件 1.0 版所定义，IAM 既不会评估 SAML 元数据文档的 X.509 证书，也不会在该证书过期时采取任何行动。如果您担心 X.509 证书过期，建议您监控证书到期日期，并根据贵组织的治理和安全策略来轮换证书。

- 选择启用 SAML 加密时，必须使用您的 IdP 生成私有密钥文件，并以 .pem 文件格式将此文件上传到 IAM SAML 配置。Amazon STS 需要此私有密钥文件来解密与 IdP 使用的公有密钥对应的 SAML 响应。支持下列算法：
  - 加密算法
    - AES-128
    - AES-256
    - RSA-OAEP
  - 密钥传输算法
    - AES-CBC
    - AES-GCM

  有关生成私有密钥的步骤，请参阅身份提供者的文档。

  > **Note**
  >
  > IAM Identity Center 和 Amazon Cognito 不支持来自 IAM SAML 身份提供商的加密 SAML 断言。您可以将对加密 SAML 断言的支持间接添加到具有 Amazon Cognito 用户池的 Amazon Cognito 身份池联合身份验证。用户池具有独立于 IAM SAML 联合身份验证的 SAML 联合身份验证，并且支持 SAML 签名和加密。尽管此功能没有直接扩展到身份池，但用户池可以是

身份池的 IDP。要对身份池使用 SAML 加密，请将具有加密的 SAML 提供商添加到一个用户池，其是身份池的 IdP。
SAML 提供商必须能够使用用户池提供的密钥对 SAML 断言进行加密。用户池不接受使用 IAM 提供的证书进行加密的断言。

有关如何配置许多可用 IdP 以使用 Amazon（包括如何生成所需的 SAML 元数据文档）的说明，请参阅将第三方 SAML 解决方案提供者与 Amazon 集成。

有关 SAML 联合身份验证的帮助，请参阅 SAML 联合身份验证故障排除。

创建和管理 IAM SAML 身份提供商（控制台）

您可以使用 Amazon Web Services 管理控制台 来创建、更新和删除 IAM SAML 身份提供商。有关 SAML 联合身份验证的帮助，请参阅 SAML 联合身份验证故障排除。

创建 IAM SAML 身份提供程序（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Identity providers（身份提供程序），然后选择 Add provider（添加提供商）。
3. 对于 Configure provider（配置提供商），选择 SAML。
4. 键入身份提供程序的名称。
5. 对于 Metadata document（元数据文档），选择 Choose file（选择文件），指定您在 the section called “先决条件” 中下载的 SAML 元数据文档。

> **Note**
>
> 针对身份提供商显示的有效截止日期由 IAM 在创建提供商时设置。它不反映 SAML 元数据文档中的 `validUntil` 或 `cacheDuration` 属性。IAM 不使用这些元数据属性来确定有效截止日期。
> 在 SAML 身份验证期间，IAM 不会对 SAML 元数据文档或该文档中的 X.509 证书过期进行评估或采取行动。使用断言的 `Conditions` 和 `SubjectConfirmationData` 元素中的 `NotOnOrAfter` 属性，在断言级别强制执行 SAML 身份验证的有效性。如果您担心 X.509 证书过期，建议您监控证书到期日期，并根据贵组织的治理和安全策略来轮换证书。

6. （可选）对于 SAML 加密，请选择选择文件并选择您在the section called “先决条件”中创建的私有密钥文件。选择需要加密，以仅接受来自您的 IdP 的加密请求。
7. （可选）对于 Add tags (添加标签)，您可以添加键值对来帮助识别和组织您的 IdP。您还可以使用标签来控制对 Amazon 资源的访问。要了解有关标记 SAML 身份提供程序的更多信息，请参阅标记 IAM SAML 身份提供者。

   选择 Add tag（添加标签）。为每个标签键值对输入值。
8. 验证您提供的信息。完成后，选择 Add provider（添加提供商）。
9. 为您的身份提供商分配一个 IAM 角色。此角色授予由您的身份提供者管理的外部用户身份访问您账户中的 Amazon 资源的权限。要了解有关为联合身份创建角色的更多信息，请参阅 为第三方身份提供者创建角色

   > **Note**
   >
   > 角色信任策略中使用的 SAML IdP 必须与角色位于同一账户中。

删除 SAML 提供商（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Identity providers（身份提供程序）。
3. 选中要删除的身份提供程序旁边的单选按钮。
4. 选择删除。此时会打开一个新窗口。
5. 通过在字段中键入 `delete` 一词以确认您要删除此提供商。然后选择 Delete(删除)。

管理 SAML 加密密钥

您可以将 IAM SAML 提供商配置为在来自外部 IdP 的 SAML 响应中接收加密断言。用户可以通过调用 `sts:AssumeRoleWithSAML` 来使用加密的 SAML 断言在 Amazon 中代入角色。

SAML 加密确保断言在通过中介或第三方传递时是安全的。此外，此功能可帮助您满足 FedRAMP 或任何要求对 SAML 断言进行加密的内部合规策略要求。

要配置 IAM SAML 身份提供商，请参阅 在 IAM 中创建 SAML 身份提供者。有关 SAML 联合身份验证的帮助，请参阅 SAML 联合身份验证故障排除。

轮换 SAML 加密密钥

IAM 使用您上传到 IAM SAML 提供商的私有密钥来解密来自 IdP 的加密 SAML 断言。每个身份提供商最多可以保存两个私钥文件，允许您根据需要轮换私钥。保存两个文件后，每个请求将首先尝试使用最新的添加日期进行解密，然后 IAM 尝试使用最旧的添加日期对请求进行解密。

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择身份提供商，然后从列表中选择您的提供商。
3. 选择 SAML 加密选项卡，然后选择添加新密钥。
4. 选择选择文件，并将您从 IdP 下载的私有密钥作为 .pem 文件上传。然后，选择添加密钥。
5. 在用于 SAML 解密的私钥部分，选择已过期的私钥文件，然后选择删除。建议您在添加新的私有密钥后删除过期的私有密钥，以确保首次尝试断言解密的成功。

创建和管理 IAM SAML 身份提供商 (Amazon CLI)

您可以使用 Amazon CLI 来创建、更新和删除 SAML 提供商。有关 SAML 联合身份验证的帮助，请参阅 SAML 联合身份验证故障排除。

创建 IAM 身份提供程序并上传元数据文档 (Amazon CLI)

- 运行以下命令：`aws iam create-saml-provider`

更新 IAM SAML 身份提供商 (Amazon CLI)

您可以更新 IAM SAML 提供商的元数据文件、SAML 加密设置并轮换私钥解密文件。要轮换私有密钥，请添加您的新私有密钥，然后在单独的请求中删除旧密钥。有关轮换私有密钥的更多信息，请参阅管理 SAML 加密密钥。

- 运行以下命令：`aws iam update-saml-provider`

要标记现有 IAM 身份提供程序 (Amazon CLI)

- 运行以下命令：`aws iam tag-saml-provider`

要列出现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：`aws iam list-saml-provider-tags`

要删除现有 IAM 身份提供程序 (Amazon CLI) 的标签

- 运行以下命令：aws iam untag-saml-provider

删除 IAM SAML 身份提供程序 (Amazon CLI)

1. （可选）要列出所有提供商的信息（例如 ARN、创建日期和过期时间），请运行以下命令：
   - aws iam list-saml-providers
2. （可选）要获取有关特定提供商的信息，如 ARN、创建日期、过期日期、加密设置和私钥信息，请运行以下命令：
   - aws iam get-saml-provider
3. 要删除 IAM 身份提供程序，请运行以下命令：
   - aws iam delete-saml-provider

创建和管理 IAM SAML 身份提供商 (Amazon API)

您可以使用 Amazon API 来创建、更新和删除 SAML 提供商。有关 SAML 联合身份验证的帮助，请参阅 SAML 联合身份验证故障排除。

创建 IAM 身份提供程序并上传元数据文档 (Amazon API)

- 调用此操作：CreateSAMLProvider

更新 IAM SAML 身份提供商 (Amazon API)

您可以更新 IAM SAML 提供商的元数据文件、SAML 加密设置并轮换私钥解密文件。要轮换私有密钥，请添加您的新私有密钥，然后在单独的请求中删除旧密钥。有关轮换私有密钥的更多信息，请参阅管理 SAML 加密密钥。

- 调用此操作：UpdateSAMLProvider

要标记现有 IAM 身份提供程序 (Amazon API)

- 调用此操作：TagSAMLProvider

要列出现有 IAM 身份提供程序 (Amazon API) 的标签

- 调用此操作：ListSAMLProviderTags

要删除现有 IAM 身份提供程序的标签 (Amazon API)

- 调用此操作：UntagSAMLProvider

删除 IAM 身份提供程序 (Amazon API)

1. （可选）要列出所有 IdP 的信息（例如 ARN、创建日期和过期时间），请调用以下操作：
   - ListSAMLProviders
2. （可选）要获取有关特定提供商的信息，如 ARN、创建日期、过期日期、加密设置和私钥信息，请调用以下操作：
   - GetSAMLProvider
3. 要删除 IdP，请调用以下操作：
   - DeleteSAMLProvider

后续步骤

创建 SAML 身份提供商后，通过您的 IdP 设置依赖方信任。您还可以在策略中使用 IdP 身份验证响应中的声明控制对角色的访问。

- 您必须告知 IdP 将 Amazon 作为服务提供商。这称为在 IdP 和 Amazon 之间添加依赖方信任。添加信赖方信任的具体步骤取决于您使用的 IdP。有关更多信息，请参阅 配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述。
- 当 IdP 将包含断言的响应发送到 Amazon 时，许多传入断言将映射到 Amazon 上下文密钥。您可以使用 Condition 元素在 IAM 策略中使用这些上下文键来控制对角色的访问权限。有关详细信息，请参阅 为身份验证响应配置 SAML 断言。

配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述

当您创建 IAM 身份提供者和用于 SAML 访问的角色时，您实际上将告知 Amazon 关于外部身份提供者 (IdP) 的信息以及允许其用户执行的操作。下一步是让 IdP 知道 Amazon 作为服务提供商。这称为

在 IdP 和 Amazon 之间添加依赖方信托。添加信赖方信任的具体步骤取决于您使用的 IdP。有关详细信息，请参阅身份管理软件对应的文档。

许多 IdP 允许指定一个 URL，他们可从中读取包含信赖方信息和证书的 XML 文档。对于 Amazon，请使用登录端点 URL。以下示例显示了包含可选 `region-code` 的 URL 格式。

```
https://region-code.signin.aws.amazon.com/static/saml-metadata.xml
```

如果需要 SAML 加密，则 URL 必须包含 Amazon 分配给您的 SAML 提供商的唯一标识符，您可以在身份提供者详细信息页面上找到该标识符。以下示例显示了包含唯一标识符的区域登录 URL。

```
https://region-code.signin.aws.amazon.com/static/saml/IdP-ID/saml-metadata.xml
```

有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。对于 Amazon 值，您也可以使用非区域端点 `https://signin.aws.amazon.com/saml` 。

如果您无法直接指定 URL，请从之前的 URL 下载 XML 文档，然后将其导入您的 IdP 软件。

您还需要在指定 Amazon 作为信赖方的 IdP 中，创建相应的声明规则。当 IdP 向 Amazon 端点发送 SAML 响应时，它包括具有一个或多个索赔 的 SAML 断言。断言是有关用户及其组的信息。断言规则将该信息映射到 SAML 属性中。这可确保来自 IdP 的 SAML 身份验证响应包含 IAM 策略中 Amazon 用于检查 SAML 联合主体权限的必要属性。有关更多信息，请参阅以下主题：

- 用于允许对 Amazon 资源进行 SAML 联合访问的角色的概述。本主题将讨论如何在 IAM 策略中使用特定于 SAML 的键以及如何使用它们限制 SAML 联合主体的权限。
- 为身份验证响应配置 SAML 断言。. 本主题将讨论如何配置包括用户相关信息的 SAML 陈述。将声明捆绑到 SAML 断言中并包括在发送到 Amazon 的 SAML 响应中。您必须确保 Amazon 策略所需的信息以 Amazon 可识别和使用的形式包括在 SAML 断言中。
- 将第三方 SAML 解决方案提供者与 Amazon 集成。本主题提供了由第三方组织提供的关于如何与 Amazon 集成身份解决方案的文档链接。

> **Note**
>
> 为了提高联合身份验证弹性，我们建议您将 IdP 和Amazon联合身份验证配置为支持多个 SAML 登录端点。有关详细信息，请参阅 Amazon 安全博客文章如何使用区域性 SAML 端点进行失效转移。

将第三方 SAML 解决方案提供者与 Amazon 集成

> **Note**
>
> 我们建议您要求您的人类用户在访问 Amazon 时使用临时凭证。您是否考虑过使用 Amazon IAM Identity Center？ 您可以使用 IAM Identity Center 集中管理对多个 Amazon Web Services 账户 的访问权限，并为用户提供受 MFA 保护的单点登录访问权限，可从一个位置访问其分配的所有账户。借助 IAM Identity Center，您可以在 IAM Identity Center 中创建和管理用户身份，或者轻松连接到现有的 SAML 2.0 兼容身份提供者。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的什么是 IAM Identity Center？。

以下链接将帮助您配置第三方 SAML 2.0 身份提供程序 (IdP) 解决方案以利用 Amazon 联合身份验证。请与您的身份提供商联系，以确定他们是否支持 SAML 令牌加密。有关 SAML 加密要求，请参阅 管理 SAML 加密密钥。

> **Tip**
>
> Amazon Support 工程师可以帮助有商业和企业支持计划的客户完成涉及第三方软件的一些集成任务。有关受支持的平台和应用程序的最新列表，请参阅《Amazon Support 常见问题》中的支持哪些第三方软件？。

解决方案	更多信息
Auth0	与 Amazon Web Services 集成 – Auth0 文档网站上的此页面包含了介绍如何使用 Amazon Web Services 管理控制台 来设置单点登录（SSO）的资源的链接，并提供了一个 JavaScript 示例。您可以配置 Auth0 来传递会话标签。有关更多信息，请参阅 Auth0 宣布与 Amazon 合作开发 IAM 会话标签。
Microsoft Entra	教程：Microsoft Entra SSO 与 Amazon 单账户访问集成 - Microsoft 网站上的这篇教程介绍了如何使用 SAML 联合身份验证将 Microsoft Entra（以前称为 Azure AD）设置为身份提供者（IdP）。

解决方案	更多信息
Centrify	Configure Centrify and Use SAML for SSO to Amazon（配置 Centrify 并对亚马逊云科技使用 SAML SSO）- 此 Centrify 网站页面说明如何配置 Centrify 以对 Amazon 使用 SAML SSO。
CyberArk	配置 CyberArk，以便为从 CyberArk 用户门户通过 SAML 单点登录(SSO)登入的用户提供 Amazon Web Services (Amazon) 访问。
ForgeRock	ForgeRock Identity Platform 与 Amazon 集成。您可以配置 ForgeRock 来传递会话标签。有关更多信息，请参阅 Amazon Web Services 的基于属性的访问控制。
Google Workspace	Amazon Web Services 云应用程序 - 这篇有关 Google Workspace Admin Help 站点的文章描述了如何将 Google Workspace 配置为 SAML 2.0 IdP，并使用 Amazon 作为服务提供程序。
IBM	您可以配置 IBM 来传递会话标签。有关更多信息，请参阅 IBM Cloud Identity IDaaS（最早支持 Amazon 会话标签的解决方案之一）。
JumpCloud	为 Amazon 单点登录（SSO）授予通过 IAM 角色进行访问的权限Amazon – 这篇有关 JumpCloud 网站的文章将描述如何为 Amazon 设置与启用基于 IAM 角色的 SSO。
Matrix42	MyWorkspace 入门指南 - 本指南介绍如何将 Amazon 身份服务与 Matrix42 MyWorkspace 集成。

解决方案	更多信息
Microsoft Active Directory 联合身份验证服务 (AD FS)	现场记录：将 Active Directory 联合身份验证服务与 Amazon IAM Identity Center 集成 – Amazon 架构博客上的这篇文章介绍了 AD FS 与 Amazon IAM Identity Center（IAM Identity Center）之间的身份验证流。IAM Identity Center 通过 SAML 2.0 支持身份联合验证，从而得以与 AD FS 解决方案集成。用户可使用其企业凭证登录 IAM Identity Center 门户，从而减少在 IAM Identity Center 上维护单独凭证的管理开销。您也可以配置 AD FS 来传递会话标签。有关更多信息，请参阅将基于属性的访问控制与 AD FS 结合使用来简化 IAM 权限管理。
miniOrange	SSO for Amazon - miniOrange 网站上的此页面介绍如何为企业建立对 Amazon 的安全访问以及对 Amazon 应用程序访问的完全控制。
Okta	使用 Okta 集成 Amazon Web Services Command Line Interface - 通过 Okta 支持网站上的此页面可以了解如何配置 Okta 才能与 Amazon 配合使用。您可以配置 Okta 来传递会话标签。有关更多信息，请参阅 Okta 与 Amazon 合作，通过会话标签简化访问。
Okta	Amazon 账户联合身份验证 - Okta 网站上的此部分介绍如何为 Amazon 设置和启用 IAM Identity Center。
OneLogin	从 OneLogin 知识库中，搜索 **SAML Amazon** 以获取文章列表，这些文章介绍了如何在 OneLogin 与 Amazon 之间设置 IAM Identity Center 功能，以实现单角色和多角色方案。您可以配置 OneLogin 来传递会话标签。有关更多信息，请参阅 OneLogin 和会话标签：Amazon 资源的基于属性的访问控制。

解决方案	更多信息
Ping Identity	PingFederate Amazon Connector – 查看有关 PingFederate Amazon Connector 的详细信息，快速连接模板来轻松设置单点登录 (SSO) 和预配置连接。阅读文档，并下载最新的 PingFederate Amazon Connector，以便与 Amazon 集成。您可以配置 Ping Identity 来传递 会话标签。有关更多信息，请参阅宣布推出针对 Amazon 中的基于属性的访问控制的 Ping 身份支持。
RadiantLogic	Radiant Logic 技术合作伙伴 - Radiant Logic 的 RadiantOne Federated Identity Service 与 Amazon 进行了集成，可以为基于 SAML 的 SSO 提供身份中心。
RSA	《Amazon Web Services - RSA Ready 实施指南》为集成 Amazon 和 RSA 提供了指导。有关 SAML 配置的更多信息，请参阅 Amazon Web Services - SAML 我的页面 SSO 配置 - RSA Ready 实施指南。
Salesforce.com	How to configure SSO from Salesforce to Amazon - Salesforce.com 开发人员站点上的这篇方法文章介绍如何在 Salesforce 中设置身份提供程序 (IdP) 以及如何将 Amazon 配置为服务提供商。
SecureAuth	Amazon - SecureAuth SAML SSO - SecureAuth 网站上的该文章介绍了如何设置 SAML 以与 SecureAuth 设备的 Amazon 集成在一起。
Shibboleth	如何使用 Shibboleth 对 Amazon Web Services 管理控制台进行 SSO - Amazon 安全博客中的这篇文章提供分步教程，介绍如何设置 Shibboleth 并将它配置为 Amazon 的身份提供程序。您可以配置 Shibboleth 来传递会话标签。

有关更多详细信息，请参阅 Amazon 网站上的 IAM 合作伙伴页面。

为身份验证响应配置 SAML 断言。

在您验证组织中的用户身份后，外部身份提供者 (IdP) 将向 Amazon 登录端点 URL 发送身份验证响应。此响应是一个包含 SAML 令牌的 POST 请求，该令牌遵循适用于 SAML 2.0 的 HTTP POST 绑定标准，其中包含以下元素或断言。您可在与 SAML 兼容的 IdP 中配置这些断言。请参考 IdP 文档，以了解有关如何输入这些声明的说明。

当 IdP 将包含断言的响应发送到 Amazon 时，许多传入断言将映射到 Amazon 上下文密钥。可以在 IIAM 策略中使用 `Condition` 元素检查这些上下文密钥。可用映射的列表如将 SAML 属性映射到 Amazon 信任策略上下文密钥部分中所示。

`Subject` 和 `NameID`

响应必须恰好具有一个 `SubjectConfirmation` 元素，该元素具有同时包含 `NotOnOrAfter` 和 `Recipient` 属性的 `SubjectConfirmationData` 元素。收件人属性必须包含与 Amazon 登录端点 URL 匹配的值。您的 IdP 可能使用 `ACS`、`Recipient`、或 `Target` 等术语来指代此属性。

如果需要 SAML 加密，则登录 URL 必须包含 Amazon 分配给您的 SAML 提供商的唯一标识符，您可以在身份提供者详细信息页面上找到该标识符。以下示例显示了带有可选 `region-code` 的登录 URL 格式。

`https://`*`region-code`*`.signin.aws.amazon.com/saml`

在以下示例中，登录 URL 包含要求在登录路径后附加 /acs/ 的唯一标识符。

`https://`*`region-code`*`.signin.aws.amazon.com/saml/acs/`*`IdP-ID`*

有关可能的 *`region-code`* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。对于 Amazon 值，您还可以使用全局登录端点 `https://signin.aws.amazon.com/saml`。

`NameID` 元素的值可以是持久性的、瞬态的或由 IdP 解决方案提供的完整格式 URI 构成。持久性值表示在不同会话之间用户的 `NameID` 值是相同的。如果值是瞬态的，则用户在每个会话中拥有不同的 `NameID` 值。单点登录交互支持以下类型的标识符：

- `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
- `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`

- urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName
- urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName
- urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos
- urn:oasis:names:tc:SAML:2.0:nameid-format:entity

以下摘录显示了一个示例。用您自己的值替代标记值。

```
<Subject>
  <NameID Format="urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent">_cbb88bf52c2510eabe00c1642d4643f41430fe25e3</NameID>
  <SubjectConfirmation Method="urn:oasis:names:tc:SAML:2.0:cm:bearer">
    <SubjectConfirmationData NotOnOrAfter="2013-11-05T02:06:42.876Z"
 Recipient="https://region-code.signin.aws.amazon.com/saml/SAMLSP4SHN3UIS2D558H46"/>
  </SubjectConfirmation>
</Subject>
```

> **Important**
>
> saml:aud 上下文键来自 SAML recipient 属性，因为它等同于 OIDC 受众字段（例如 accounts.google.com:aud）的 SAML。

PrincipalTag SAML 属性

（可选）您可使用将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/PrincipalTag:*{TagKey}* 的 Attribute 元素。此元素允许您将属性作为 SAML 断言中的会话标签传递。有关会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。

要将属性作为会话标签传递，请包含指定标签值的 AttributeValue 元素。例如，要传递标签键/值对 Project = Marketing 和 CostCenter = 12345，请使用以下属性。为每个标签包含一个单独的 Attribute 元素。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Marketing</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
```

要将上述标签设置为可传递，请包含另一个 Attribute 元素并将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys。这是一个可选的多值属性，可将您的会话标签设置为可传递。当您使用 SAML 会话代入 Amazon 中的另一个角色时，可传递标签将保留。这称为角色链。例如，要将 Principal 和 CostCenter 标签均设置为可传递，请使用以下属性指定键。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>CostCenter</AttributeValue>
</Attribute>
```

Role SAML 属性

您可使用将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/Role 的 Attribute 元素。此元素包含一个或多个 AttributeValue 元素，这些元素可列出用户通过 IdP 映射到的 IAM 身份提供者和角色。IAM 角色和 IAM 身份提供者指定为逗号分隔的 ARN 对，格式与传递到 AssumeRoleWithSAML 的 RoleArn 和 PrincipalArn 参数相同。此元素必须至少包含一个角色/提供商对（AttributeValue 元素），也可以包含多个对。如果该元素包含多个对，则用户需要选择其使用 WebSSO 登录 Amazon Web Services 管理控制台 时要代入的角色。

> **Important**
>
> Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 https://aws.amazon.com/SAML/Attributes/Role。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/Role">
  <AttributeValue>arn:aws:iam::account-number:role/role-name1,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name2,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
  <AttributeValue>arn:aws:iam::account-number:role/role-name3,arn:aws:iam::account-number:saml-provider/provider-name</AttributeValue>
</Attribute>
```

RoleSessionName SAML 属性

您可使用将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/RoleSessionName 的 Attribute 元素。该元素包含一个为临时凭证提供标识符的

AttributeValue 元素，这些凭证是针对 SSO 颁发的。您可以使用此选项将临时凭证与正在使用您应用程序的用户相关联。该元素用于在 Amazon Web Services 管理控制台中显示用户信息。AttributeValue 元素中的值长度必须介于 2 到 64 个字符之间，只能包含字母数字字符、下划线和以下字符：. , + = @ - (连字符) 。它不能含有空格。该值通常是用户 ID (john) 或电子邮件地址 (johndoe@example.com)。该值不应包含空格，如用户的显示名称 (John Doe)。

> **Important**
>
> Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 https://aws.amazon.com/SAML/Attributes/RoleSessionName。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/RoleSessionName">
  <AttributeValue>user-id-name</AttributeValue>
</Attribute>
```

SessionDuration SAML 属性

(可选) 您可使用将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/SessionDuration 的 Attribute 元素。该元素包含一个 AttributeValue 元素，它指定用户在必须请求新的临时凭证之前可访问 Amazon Web Services 管理控制台的时间长度。该值是一个表示会话秒数的整数。该值的范围是 900 秒 (15 分钟) 到 43200 秒 (12 小时)。如果该属性不存在，则凭证持续时间为 1 小时 (DurationSeconds API 的 AssumeRoleWithSAML 参数的默认值)。

要使用该属性，您必须配置 SAML 提供商，以通过控制台登录 Web 端点 (https://*region-code*.signin.aws.amazon.com/saml) 提供对 Amazon Web Services 管理控制台的单一登录访问。有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region (区域) 列。您可以选择使用以下 URL：https://signin.aws.amazon.com/static/saml 。请注意，该属性仅将会话扩展到 Amazon Web Services 管理控制台。它不能延长其他凭证的有效期。但是，如果它存在于 AssumeRoleWithSAML API 调用中，则可以用来缩短 会话的持续时间。调用返回的凭证的默认生命周期为 60 分钟。

另请注意，如果还定义了 SessionNotOnOrAfter 属性，则两个属性的 lesser (较小值) (SessionDuration 或 SessionNotOnOrAfter) 将建立控制台会话的最大持续时间。

在启用具有更长持续时间的控制台会话时，可能产生凭证外泄的风险。为了帮助缓解这种风险，您可以通过在 IAM 控制台页面的 Role Summary (角色摘要) 上选择 Revoke Sessions (撤销会话) 立即禁用所有角色的有效控制台会话。有关更多信息，请参阅 撤销 IAM 角色临时安全凭证。

> **Important**
>
> Name 标签中 Attribute 属性的值区分大小写。必须将其一字不差地设置为 https://aws.amazon.com/SAML/Attributes/SessionDuration。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SessionDuration">
  <AttributeValue>1800</AttributeValue>
</Attribute>
```

SourceIdentity SAML 属性

（可选）您可使用将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/SourceIdentity 的 Attribute 元素。该元素包含一个 AttributeValue 元素，为使用 IAM 角色的人员或应用程序提供标识符。当您使用 SAML 会话代入 Amazon 中的另一个角色时，源身份的值将保留，称为角色链。源身份的值存在于对角色会话期间执行的每个操作的请求中。在角色会话期间无法更改已设置的值。然后，管理员可以使用 Amazon CloudTrail 日志来监控和审计源身份信息，以确定谁在使用共享角色执行操作。

AttributeValue 元素中的值长度必须介于 2 到 64 个字符之间，只能包含字母数字字符、下划线和以下字符：. , + = @ -（连字符）。它不能含有空格。该值通常是与用户关联的属性，例如用户 ID (john) 或电子邮件地址 (johndoe@example.com)。该值不应包含空格，如用户的显示名称 (John Doe)。有关使用基于身份的策略的更多信息，请参阅 监控和控制使用所担任角色执行的操作。

> **Important**
>
> 如果您的 SAML 断言配置为使用 SourceIdentity 属性，则您的角色信任策略还必须包含 sts:SetSourceIdentity 操作，否则代入角色操作会失败。有关使用基于身份的策略的更多信息，请参阅 监控和控制使用所担任角色执行的操作。

要传递源身份属性，请包含 AttributeValue 元素，指定源身份的值。例如，要传递源身份 Diego 使用以下属性。

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
  <AttributeValue>Diego</AttributeValue>
```

将 SAML 属性映射到 Amazon 信任策略上下文密钥

本部分中的表列出了常用的 SAML 属性以及它们在 Amazon 中映射到信任策略条件上下文密钥的方式。您可以使用这些键来控制对角色的访问。为此，请将键与 SAML 访问请求中所附带断言包含的值进行比较。

> **Important**
>
> 这些键仅在 IAM 信任策略（可决定谁能担任角色的策略）中可用，并且不适用于权限策略。

在 eduPerson 和 eduOrg 属性表中，值是以字符串或字符串列表的形式键入的。对于字符串值，您可以使用 `StringEquals` 或 `StringLike` 条件测试 IAM 信任策略中的这些值。对于包含字符串列表的值，您可以使用 `ForAnyValue` 和 `ForAllValues` 策略集运算符测试信任策略中的这些值。

> **Note**
>
> 每个 Amazon 上下文密钥只能包含一个断言。如果包含多个声明，将仅映射其中一个。

下表显示 eduPerson 和 eduOrg 属性。

eduPerson 或 eduOrg 属性（**`Name`** 密钥）	映射至此 Amazon 上下文密钥（**`FriendlyName`** 密钥）	Type
`urn:oid:1.3.6.1.4.1.5923.1.1.1.1`	`eduPerson Affiliation`	字符串列表
`urn:oid:1.3.6.1.4.1.5923.1.1.1.2`	`eduPersonNickname`	字符串列表
`urn:oid:1.3.6.1.4.1.5923.1.1.1.3`	`eduPersonOrgDN`	字符串
`urn:oid:1.3.6.1.4.1.5923.1.1.1.4`	`eduPerson OrgUnitDN`	字符串列表
`urn:oid:1.3.6.1.4.1.5923.1.1.1.5`	`eduPerson PrimaryAf filiation`	字符串

eduPerson 或 eduOrg 属性（**Name** 密钥）	映射至此 Amazon 上下文密钥（**FriendlyName** 密钥）	Type
urn:oid:1.3.6.1.4.1.5923.1.1.1.6	eduPerson PrincipalName	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.7	eduPerson Entitlement	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.1.1.8	eduPerson PrimaryOrgUnitDN	字符串
urn:oid:1.3.6.1.4.1.5923.1.1.1.9	eduPerson ScopedAff iliation	字符串列表
urn:oid:1.3.6.1.4.1.5923.1. 1.1.10	eduPerson TargetedID	字符串列表
urn:oid:1.3.6.1.4.1.5923.1. 1.1.11	eduPerson Assurance	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.2	eduOrgHomePageURI	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.3	eduOrgIde ntityAuth NPolicyURI	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.4	eduOrgLegalName	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.5	eduOrgSuperiorURI	字符串列表
urn:oid:1.3.6.1.4.1.5923.1.2.1.6	eduOrgWhi tePagesURI	字符串列表
urn:oid:2.5.4.3	cn	字符串列表

下表显示 Active Directory 属性。

AD 属性	映射到此 Amazon 上下文密钥	Type
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	name	字符串
http://schemas.xmlsoap.org/claims/CommonName	commonName	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/givenname	givenName	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/surname	surname	字符串
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress	mail	字符串
http://schemas.microsoft.com/ws/2008/06/identity/claims/primarygroupsid	uid	字符串

下表显示 X.500 属性。

X.500 属性	映射到此 Amazon 上下文密钥	Type
2.5.4.3	commonName	字符串
2.5.4.4	surname	字符串
2.4.5.42	givenName	字符串
2.5.4.45	x500UniqueIdentifier	字符串
0.9.2342.19200300100.1.1	uid	字符串
0.9.2342.19200300100.1.3	mail	字符串

X.500 属性	映射到此 Amazon 上下文密钥	Type
`0.9.2342.19200300.100.1.45`	`organizationStatus`	字符串

使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台

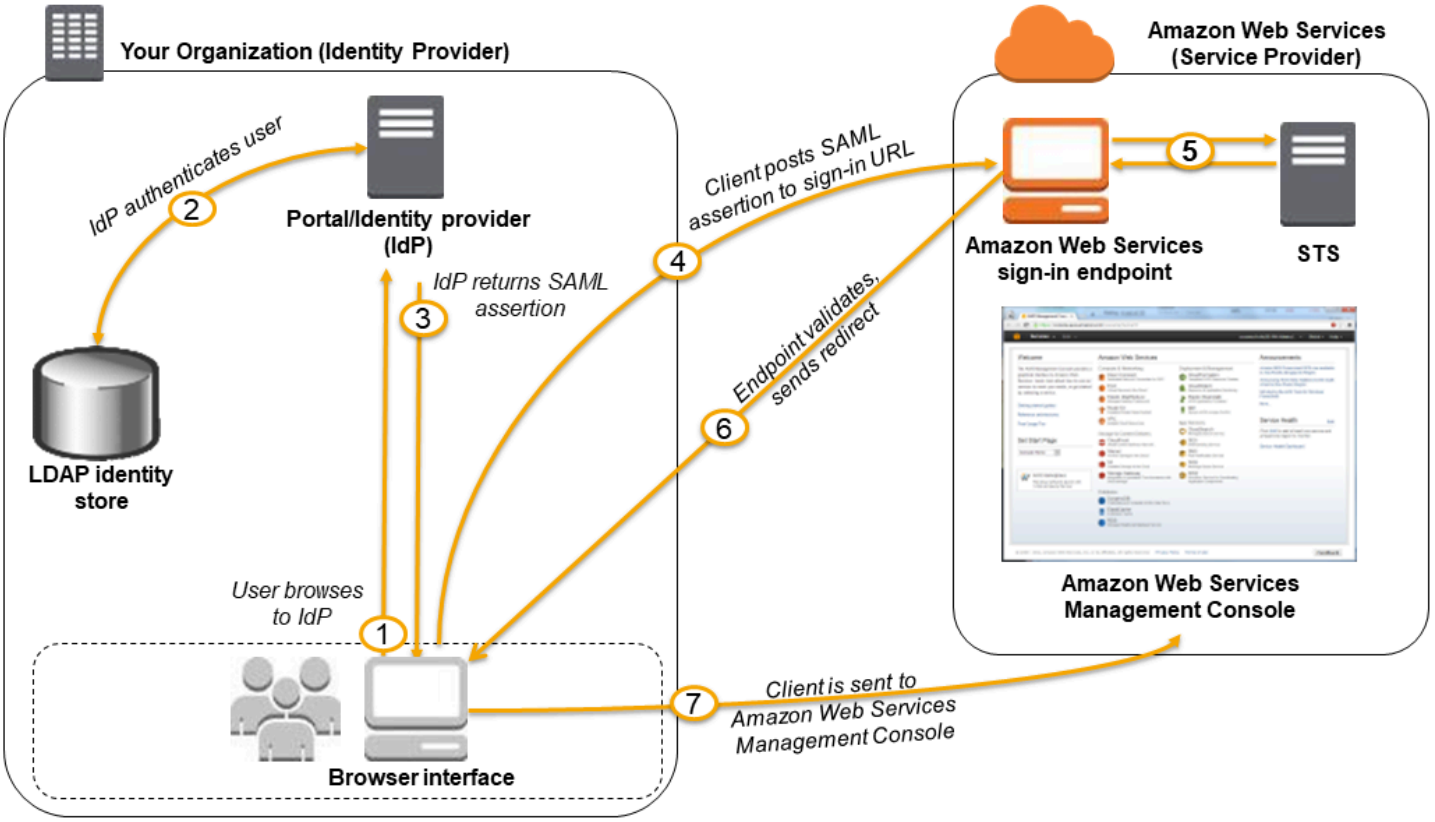
您可使用一个角色配置符合 SAML 2.0 标准的身份提供者 (IdP) 和 Amazon 以允许 SAML 联合主体访问 Amazon Web Services 管理控制台。该角色为用户授予了在控制台中执行任务的权限。如果您希望为 SAML 联合主体提供访问 Amazon 的其他方式，请参阅以下其中一项主题：

- Amazon CLI: 切换到 IAM 角色（Amazon CLI）
- Tools for Windows PowerShell: 切换到 IAM 角色（Tools for Windows PowerShell）
- Amazon API：切换到 IAM 角色（Amazon API）

概述

下图说明了启用了 SAML 的单一登录的流程。

> **Note**
>
> SAML 的这种特定用途与SAML 2.0 联合身份验证中所述的更常规的用途不同，因为此工作流程将代表用户打开 Amazon Web Services 管理控制台。这需要使用 Amazon 登录端点，而不是直接调用 `AssumeRoleWithSAML` API。该端点将为用户调用 API 并返回将用户的浏览器自动重定向到 Amazon Web Services 管理控制台的 URL。



下图说明了以下步骤：

1. 用户浏览到您的组织的门户网站，并选择转到 Amazon Web Services 管理控制台的选项。在企业中，门户网站通常充当处理您企业与 Amazon 之间的信任交换的 IdP。例如，在 Active Directory 联合身份验证服务中，门户网站 URL 为：`https://ADFSServiceName/adfs/ls/IdpInitiatedSignOn.aspx`
2. 该门户网站可验证您的组织用户的身份。
3. 该门户网站生成一个 SAML 身份验证响应，其中包括识别用户身份的断言以及用户的相关属性。您也可以配置 IdP 以包含一个名为 `SessionDuration` 的 SAML 断言属性，该属性指定控制台会话的有效时间长度。您还可以配置 IdP，将属性作为会话标签传递。该门户网站将此响应发送到客户端浏览器。
4. 该客户端浏览器将被重定向到 Amazon 单一登录端点并发布 SAML 断言。
5. 端点将代表用户请求临时安全凭证，并创建一个使用这些凭证的控制台登录 URL。
6. Amazon 将登录 URL 作为重定向发回客户端。
7. 该客户端浏览器将重定向到 Amazon Web Services 管理控制台。如果 SAML 身份验证响应包含映射到多个 IAM 角色的属性，则系统将首先提示用户选择角色以访问控制台。

从用户的角度来看，整个流程以透明的方式进行：用户在您的企业的内部门户网站开始操作，在 Amazon Web Services 管理控制台 结束操作，无需提供任何 Amazon 凭证。

有关如何配置此行为的概述以及指向详细步骤的链接，请参阅以下章节。

将网络配置为适用于 Amazon 的 SAML 提供商

在组织的网络中，配置身份存储（例如 Windows Active Directory）以使用基于 SAML 的 IdP，例如 Windows Active Directory 联合身份验证服务、Shibboleth 等。通过使用 IdP，可以生成一个元数据文档，此文档将您的组织描述为 IdP 并且包含身份验证密钥。另外，还要将企业的门户网站配置为将访问 Amazon Web Services 管理控制台 的用户请求路由至 Amazon SAML 端点，以便使用 SAML 断言进行身份验证。如何配置您的 IdP 来生成 metadata.xml 文件取决于您的 IdP。请参阅您的 IdP 文档以获得指示，或参阅将第三方 SAML 解决方案提供者与 Amazon 集成以获得指向很多支持的 SAML 提供商 Web 文档。

在 IAM 中创建 SAML 提供商

然后，登录 Amazon Web Services 管理控制台并转至 IAM 控制台。在此，您创建一个新的 SAML 提供商，这是在 IAM 中包含您的企业的 IdP 的相关信息的实体。在此过程中，您可以上传在上一节中由贵组织中的 IdP 软件生成的元数据文档。有关更多信息，请参阅 在 IAM 中创建 SAML 身份提供者。

在 Amazon 中为 SAML 联合主体中配置权限

下一步是创建一个 IAM 角色，以在 IAM 与您企业的 IdP 之间建立信任关系。该角色必须将您的 IdP 标识为主体（可信实体）以实现联合身份验证目的。该角色还定义了由您的组织的 IdP 进行身份验证的用户可以在 Amazon 中执行的操作。您可使用 IAM 控制台创建该角色。在创建指示谁可以代入角色的信任策略时，您可以将先前创建的 SAML 提供商指定为 IAM。您还可以指定用户要代入角色必须匹配的一个或多个 SAML 属性。例如，您可以指定只允许其 SAML `eduPersonOrgDN` 值为 `ExampleOrg` 的用户登录。角色向导会自动添加一个测试 `saml:aud` 属性的条件，以确保仅出于登录 Amazon Web Services 管理控制台的目的担任该角色。

如果需要 SAML 加密，则登录 URL 必须包含 Amazon 分配给您的 SAML 提供商的唯一标识符，您可以在身份提供者详细信息页面上找到该标识符。该角色的信任策略可能如下所示：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Principal": {
                "Federated": "arn:aws:iam::111122223333:saml-provider/
ExampleOrgSSOProvider"
            },
            "Action": "sts:AssumeRoleWithSAML",
            "Condition": {
                "StringEquals": {
                    "saml:edupersonorgdn": "ExampleOrg",
                    "saml:aud": "https://region-code.signin.aws.amazon.com/saml/
acs/SAMLSP4SHN3UIS2D558H46"
                }
            }
        }
    ]
}
```

> **Note**
>
> 角色信任策略中使用的 SAML IdP 必须与角色位于同一账户中。

我们建议使用区域端点作为 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 的 `saml:aud` 属性。有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。

对于该角色中的权限策略，您可以像任何角色、用户或组一样指定权限。例如，如果允许您的企业中的用户管理 Amazon EC2 实例，您可以在权限策略中显式允许 Amazon EC2 操作。您可通过分配一个托管策略 (如 Amazon EC2 完全访问托管策略) 执行此操作。

有关创建用于 SAML IdP 的角色的详细信息，请参阅创建用于 SAML 2.0 联合身份验证的角色（控制台）。

完成配置并创建 SAML 断言

通过安装位于 `https://region-code.signin.aws.amazon.com/static/saml-metadata.xml` 或 `https://signin.aws.amazon.com/static/saml-metadata.xml` 的 `saml-metadata.xml` 文件，以通知 SAML IdP Amazon 是您的服务提供商。如果需要 SAML 加密，则可在 `https://region-code.signin.aws.amazon.com/static/saml/SAMLSP4SHN3UIS2D558H46/saml-metadata.xml` 查找该文件。

有关可能的 *region-code* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。

安装该文件的方式取决于您的 IdP。一些提供商为您提供了键入该 URL 的选项，此时，IdP 将为您获取并安装该文件。另一些提供商则要求您从该 URL 处下载该文件，然后将其作为本地文件提供。请参阅您的 IdP 文档以获得详细信息或参阅将第三方 SAML 解决方案提供者与 Amazon 集成以获得指向很多支持的 SAML 提供商的 Web 文档。

您还可配置一条信息，您希望 IdP 在身份验证响应期间将此信息作为 SAML 属性传递到 Amazon。这些信息的大部分在 Amazon 中显示为条件上下文密钥，您可以在策略中进行评估。这些条件键确保仅向正确上下文中的授权用户授予权限来访问您的 Amazon 资源。您可以指定限制何时可使用控制台的时段。您还可以指定在必须刷新用户的凭证之前用户可访问控制台的最长时间（最多 12 个小时）。有关详细信息，请参阅 为身份验证响应配置 SAML 断言。。

在您的浏览器中查看 SAML 响应

以下过程介绍了在排除 SAML 2.0 相关问题时，如何在浏览器中查看服务提供商的 SAML 响应。

对于所有浏览器，请转到您可以重现问题的页面。然后，针对相应的浏览器执行以下步骤：

主题

- Google Chrome
- Mozilla Firefox
- Apple Safari
- 如何处理 Base64 编码的 SAML 响应

Google Chrome

在 Chrome 中查看 SAML 响应

这些步骤已经过 Google Chrome 版本 106.0.5249.103（官方版本）（arm64）的测试。如果您使用其他版本，则可能需要相应地调整步骤。

1. 按 F12 以启动 Developer Tools（开发人员工具）控制台。
2. 选择 Network（网络）选项卡，然后选择 Developer Tools（开发人员工具）窗口左上角的 Preserve log（保留日志）。
3. 重现问题。
4. （可选）如果 Method（方法）列在 Developer Tools（开发人员工具）的 Network（网络）日志窗格中不可见，右击任何列标签并选择 Method（方法）以添加列。

5. 在 Developer Tools（开发人员工具）的 Network（网络）日志窗格中查找 SAML Post。选择该行，然后查看顶部的 Payload（有效负载）选项卡。查找包含编码请求的 SAMLResponse 元素。关联值为 Base64 编码的响应。

Mozilla Firefox

在 Firefox 中查看 SAML 响应

此过程已在 Mozilla Firefox 版本 105.0.3（64 位）上进行了测试。如果您使用其他版本，则可能需要相应地调整步骤。

1. 按 F12 以启动 Web Developer Tools（Web 开发人员工具）控制台。
2. 选择 Network 选项卡。
3. 在 Web Developer Tools（Web 开发人员工具）窗口的右上角，选择选项（小齿轮图标）。选择 Persist logs（保留日志）。
4. 重现问题。
5. （可选）如果 Method（方法）列在 Web 开发人员工具的 Network（网络）日志窗格中不可见，右击任何列标签并选择 Method（方法）以添加列。
6. 在表中查找 POST SAML。选择该行，然后查看 Request（请求）选项卡并找到 SAMLResponse 元素。关联值为 Base64 编码的响应。

Apple Safari

在 Safari 中查看 SAML 响应

这些步骤已经过 Apple Safari 版本 16.0（17614.1.25.9.10、17614）的测试。如果您使用其他版本，则可能需要相应地调整步骤。

1. 在 Safari 中启用 Web Inspector。打开 Preferences 窗口，选择 Advanced 选项卡，然后选择 Show Develop menu in the menu bar。
2. 现在您可以打开 Web Inspector。在菜单栏中选择 Develop（开发），然后选择 Show Web Inspector（显示 Web 检查器）。
3. 选择 Network 选项卡。
4. 在 Web Inspector（Web 检查器）窗口的左上角，选择选项（包含三条横线的小圆圈图标）。选择 Preserve Log（保留日志）。

5. （可选）如果 Method（方法）列在 Web Inspector（Web 检查器）的Network（网络）日志窗格中不可见，右击任何列标签并选择 Method（方法）以添加列。
6. 重现问题。
7. 在表中查找 POST SAML。选择该行，然后查看 Headers（标头）选项卡。
8. 查找包含编码请求的 SAMLResponse 元素。向下滚动，查找名为 `Request Data` 的 `SAMLResponse`。关联值为 Base64 编码的响应。

如何处理 Base64 编码的 SAML 响应

在浏览器中找到 Base64 编码的 SAML 响应元素之后，复制这些元素并使用您偏好的 Base-64 解码工具来提取带有 XML 标签的响应。

> **安全提示**
>
> 由于您查看的 SAML 响应数据可能包含敏感安全数据，我们建议您不要使用在线 base64 解码程序。而是使用安装在本地计算机上、不会通过网络发送 SAML 数据的工具。

适用于 Windows 系统的内置选项 (PowerShell)：

```
PS C:
\> [System.Text.Encoding]::UTF8.GetString([System.Convert]::FromBase64String("base64encodedtext
```

适用于 MacOS 和 Linux 系统的内置选项：

```
$ echo "base64encodedtext" | base64 --decode
```

查看解码后的文件中的值

查看解码后的 SAML 响应文件中的值。

- 验证 saml:NameID 属性的值是否与经过身份验证的用户的用户名匹配。
- 查看 https://aws.amazon.com/SAML/Attributes/Role 的值。ARN 和 SAML 提供商区分大小写，并且 ARN 必须与您账户中的资源匹配。
- 查看 https://aws.amazon.com/SAML/Attributes/RoleSessionName 的值。该值必须与声明规则中的值匹配。

- 如果为电子邮件地址或账户名配置了属性值，请确保这些值正确无误。这些值必须与经过身份验证的用户的电子邮件地址或账户名相对应。

检查是否有错并确认配置

检查这些值是否包含错误，并确认下面的配置是否正确。

- 声明规则符合要求的要素，并且所有 ARN 都正确无误。有关更多信息，请参阅 配置具有依赖方信任的 SAML 2.0 IdP 并添加陈述。
- 您已将最新元数据文件从 IdP 上传到 SAML 提供商中的 Amazon。有关更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。
- 您正确配置了 IAM 角色的信任策略。有关更多信息，请参阅 担任角色的方法。

将 Amazon 身份联合到外部服务

IAM 外站身份联合验证使您的 Amazon 工作负载无需存储长期凭证即可安全访问外部服务。您的 Amazon 工作负载可通过调用 GetWebIdentityToken API 从 Amazon Security Token Service（Amazon STS）请求短期的 JSON Web 令牌（JWT）。这些令牌经过加密签名，可公开验证，并包含一组全面的声明，用于向外部服务声明 Amazon 工作负载的身份。您可以将这些令牌用于各种第三方云提供商、SaaS 平台和自托管应用程序。外部服务使用在知名端点发布的 Amazon 验证密钥来验证令牌的真实性，并使用令牌中的信息进行身份验证和授权决策。

外站身份联合验证无需在应用程序代码或环境变量中存储 API 密钥或密码等长期凭证，从而改善了您的安全状况。您可以使用 IAM 策略控制对令牌生成的访问权限并强制执行签名算法、允许的受众和持续时间等令牌属性。所有令牌请求均已在 Amazon 中记录，从而为安全监控与合规性报告提供完整的审计跟踪记录。您还可以使用显示为自定义声明的标签自定义令牌，从而使外部服务能够实现基于属性的精细访问控制。

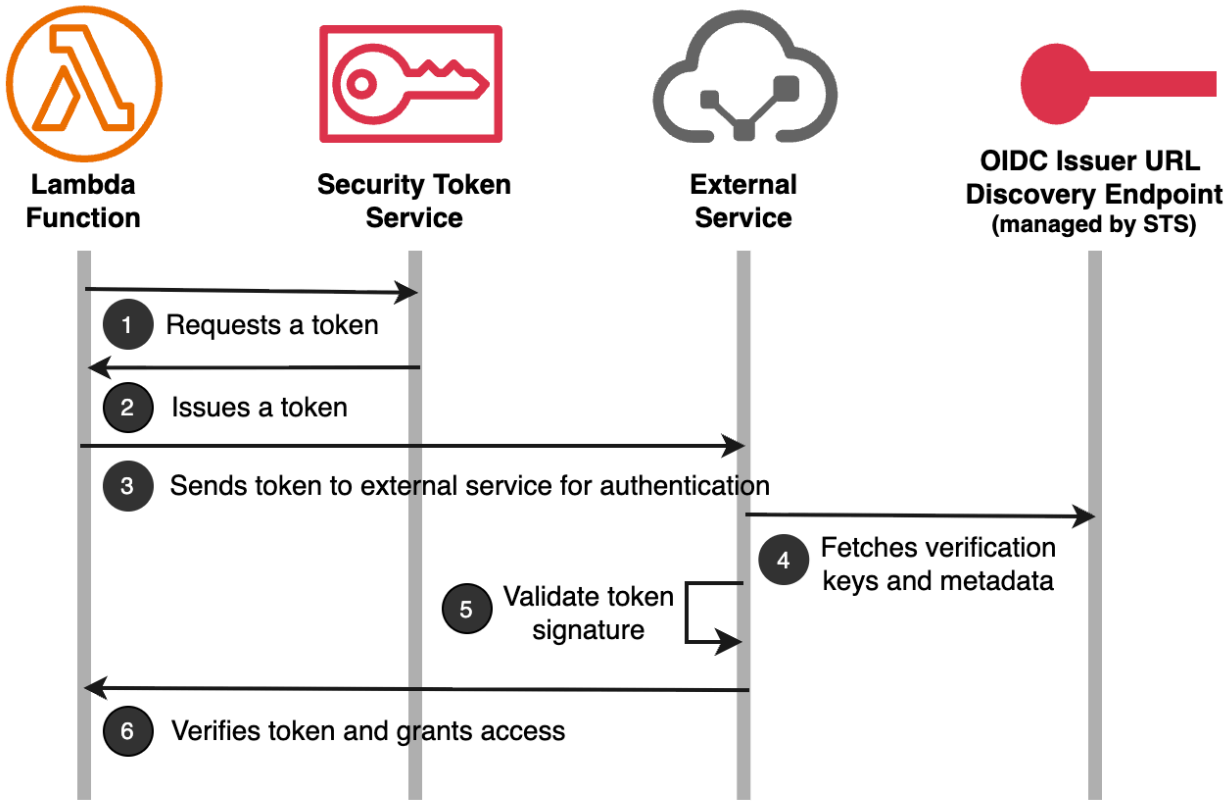
常见使用案例

使用出站身份联合验证，您的 Amazon 工作负载可以安全地实现以下功能：

- 访问外部云提供商中的资源和服务。例如，处理数据的 Lambda 函数可以将结果写入外部云提供商的存储服务或查询其数据库。
- 与外部软件即服务（SaaS）提供商集成，以进行分析、数据处理、监控等。例如，您的 Lambda 函数可以向可观测性平台发送指标。

- 使用托管在 Amazon、外部云提供商或本地数据中心上的自有应用程序进行身份验证，从而实现安全的混合云和多云架构。例如，您的 Amazon 工作负载可以与在本地 Kubernetes 集群中运行的容器化应用程序进行交互。

工作方式



1. Lambda 函数调用 GetWebIdentityToken API 向 Amazon Security Token Service（Amazon STS）请求 JSON Web 令牌（JWT）。
2. Amazon STS 验证请求并向 Lambda 函数返回已签名的 JWT。
3. Lambda 函数将 JWT 发送到外部服务。
4. 外部服务从令牌中提取发布者 URL，验证其是否与已知的可信发布者匹配，并从 OIDC 发现端点获取 Amazon 的验证密钥和元数据。
5. 外部服务使用验证密钥来验证令牌的签名，并验证过期时间、主题和受众等声明。
6. 成功验证后，外部服务将授予对 Lambda 函数的访问权限。

开始使用出站身份联合验证

本指南向您展示如何为您的 Amazon 账户启用外站身份联合验证以及如何获取您的第一个 JSON Web 令牌（JWT）（使用 GetWebIdentityToken API）。您将启用该功能，与外部服务建立信任关系，配置 IAM 权限，并使用 Amazon CLI 或适用于 Python 的 Amazon SDK（Boto3）请求令牌。

先决条件

在开始之前，请确保您满足以下条件：

- 已安装最新版本的 Amazon CLI 或 Python 3.8（或更高版本）和 Boto3（适用于 Amazon SDK 示例）
- 可以在其中配置信任关系的外部服务帐户（例如外部云提供商、SaaS 提供商或测试应用程序）

> **Note**
>
> - `GetWebIdentityToken` API 在 STS 全局端点上不可用。
> - `GetWebIdentityToken` API 生成的 JSON Web 令牌（JWT）不能用于通过 OpenID Connect（OIDC）联合身份验证登录 Amazon（通过 AssumeRoleWithWebIdentity API）。

为您的账户启用外站身份联合验证

必须先启用外站身份联合验证，然后才能请求令牌。您可以使用 Amazon 管理控制台启用该功能，也可以使用 EnableOutboundWebIdentityFederation API 以编程方式启用该功能。

使用 Amazon CLI

```
aws iam enable-outbound-web-identity-federation
```

使用适用于 Python 的 Amazon SDK

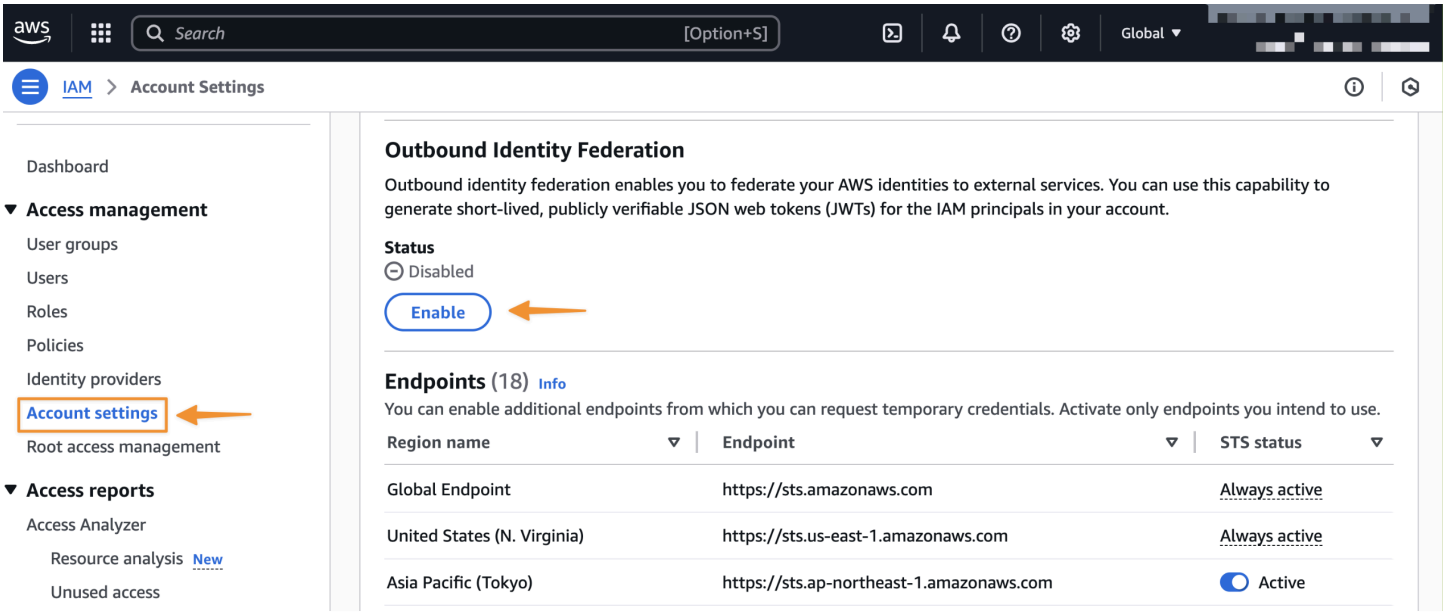
```
import boto3

# Create IAM client
iam_client = boto3.client('iam')
```

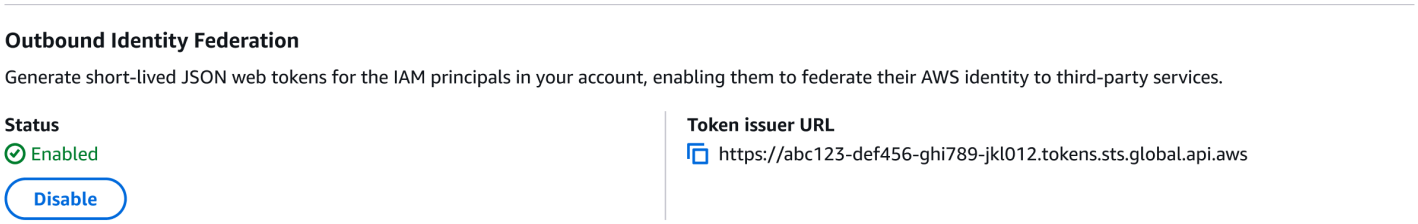
```
# Enable outbound identity federation
response = iam_client.enable_outbound_web_identity_federation()
print(f"Feature enabled. Issuer URL: {response['IssuerUrl']}")
print(f"Status: {response['Status']}")
```

使用 Amazon 控制台

导航到 IAM，然后在左侧导航菜单的访问管理部分下选择账户设置



启用该功能后，请记下您的账户特定发布者 URL。在外部服务中配置信任关系时，您将使用此 URL。您也可以根据需要使用 GetOutboundWebIdentityFederationInfo API 检索此发布者 URL。



在外部服务中建立信任关系

将外部服务配置为信任并接受由您的 Amazon 账户发布的令牌。具体步骤因服务而有所不同，但通常包括：

- 将您的 Amazon 账户发布者 URL 注册为可信身份提供商
- 配置要验证的声明（受众、主题模式）
- 将令牌声明映射到外部服务中的权限

有关详细的配置说明，请参阅外部服务文档。

配置 IAM 权限

创建一个 IAM 策略，授予调用 GetWebIdentityToken API 的权限，然后将该策略附加到需要生成令牌的 IAM 角色。

此示例策略授予生成令牌的权限，但具有特定限制。其仅允许为受众“https://api.example.com”请求令牌，并且强制规定最长令牌生命周期为 5 分钟（300 秒）。有关可用于强制执行令牌属性的条件键列表，请参阅 the section called “IAM 条件键”。

示例 IAM 策略

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:GetWebIdentityToken",
            "Resource": "*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "sts:IdentityTokenAudience": "https://api.example.com"
                },
                "NumericLessThanEquals": {
                    "sts:DurationSeconds": 300
                }
            }
        }
    ]
}
```

请求您的第一个 JSON Web 令牌（JWT）

您可以使用 GetWebIdentityToken API 请求 JSON Web 令牌。调用 API 时可以指定以下参数：

- 受众（必填）：令牌的预期接收者。该值填充 JWT 中的“aud”声明。外部服务验证此声明，以确保令牌是为其准备的。
- SigningAlgorithm（必填）：用于对令牌进行签名的加密算法。有效值为 ES384 和 RS256。使用 ES384 可获得最佳安全性和性能，或使用 RS256 可实现与不支持 ECDSA 的系统更广泛的兼容性。
- DurationSeconds（可选）：令牌生命周期（以秒为单位）。有效值范围为 60 至 3600。默认值为 300（5 分钟）。我们建议缩短令牌生命周期以提高安全性。

- 标签（可选）：要作为自定义声明包含在令牌中的键值对列表。外部服务可以使用这些声明进行精细授权。

该 API 会返回以下字段：

- IdentityToken：已签名的 JWT，以 base64url 编码字符串的形式表示。将此令牌包含在对外部服务的请求中。
- 过期：令牌过期时的 UTC 时间戳。

使用 Amazon CLI

```
aws sts get-web-identity-token \
    --audience "https://api.example.com" \
    --signing-algorithm ES384 \
    --duration-seconds 300 \
    --tags Key=team,Value=data-engineering \
           Key=environment,Value=production \
           Key=cost-center,Value=analytics
```

使用适用于 Python 的 Amazon SDK

```
import boto3

sts_client = boto3.client('sts')

response = sts_client.get_web_identity_token(
    Audience=['https://api.example.com'],
    DurationSeconds=300,
    SigningAlgorithm='RS256',
    Tags=[
        {'Key': 'team', 'Value': 'data-engineering'},
        {'Key': 'environment', 'Value': 'production'},
        {'Key': 'cost-center', 'Value': 'analytics'}
    ]
)

token = response['WebIdentityToken']
```

您也可以使用标准 JWT 库（例如 PyJWT、Python-jose for Python、Nimbus JOSE+JWT for Java 或 jwt.io 之类的调试程序）解码 JWT 以检查其内容。有关令牌中包含的声明的更多信息，请参阅the section called “了解令牌声明”。

将令牌与外部服务一起使用

收到令牌后，将其包含在对外部服务的请求中。方法因服务而有所不同，但大多数服务都接受 Authorization 标头中的令牌。外部服务应实现令牌验证逻辑，以在授予对 Amazon 工作负载的访问权限之前，从发布者的知名端点获取 JWKS 密钥，验证令牌的签名并验证基本声明。

从 OpenID Connect（OIDC）端点获取验证密钥和元数据

您 Amazon 账户唯一的发布者 URL 可托管 OpenID Connect（OIDC）发现端点，其中包含令牌验证所需的验证密钥和元数据。

OIDC 发现端点 URL 包含一些提供商用于验证令牌的元数据。网址为：

```
{issuer_url}/.well-known/openid-configuration
```

JWKS（JSON Web 密钥集）端点包含用于验证令牌签名的密钥。网址为：

```
{issuer_url}/.well-known/jwks.json
```

使用 curl 获取 JWKS

```
curl https://{issuer_url}/.well-known/jwks.json
```

响应：

```
{
  "keys": [
    {
      "kty": "EC",
      "use": "sig",
      "kid": "key-id-1",
      "alg": "ES384",
      "crv": "P-384",
      "x": "base64-encoded-x-coordinate",
      "y": "base64-encoded-y-coordinate"
```

```
    },
    {
      "kty": "RSA",
      "use": "sig",
      "kid": "key-id-2",
      "n": "base64-encoded-modulus",
      "e": "AQAB"
    }
  ]
}
```

使用适用于 Python 的 Amazon SDK

```
import requests

# Fetch Openid Configuration
open_id_config_response = requests.get("https://{issuer_url}/.well-known/openid-
configuration")
open_id_config = open_id_config_response.json()

# Fetch JWKS
jwks_response = requests.get("https://{issuer_url}/.well-known/jwks.json")
jwks = jwks_response.json()
```

我们建议缓存这些密钥，以避免每次验证令牌时都需要重新获取。

基本声明验证

- 主题（sub）：验证主题声明是否包含预期的 IAM 主体 ARN 模式。
- 过期（exp）：确保令牌未过期。JWT 库通常会自动处理此问题。
- 受众（aud）：验证受众是否匹配您的预期值。这样可以防止原本用于其他服务的令牌与您的令牌混用。
- 发布者（iss）：验证发布者是否匹配您信任的 Amazon 账户。维护可信发布者 URL 列表。

应尽可能验证其他 Amazon 特定声明，以在外部服务中实现精细访问控制。例如，验证 org_id 声明以限制 Amazon Organization 中 IAM 主体的访问权限，选中 principal_tags 以强制执行基于属性的访问控制（例如仅允许生产环境或特定团队），或者验证 lambda_source_function_arn 或 ec2_instance_source_vpc 等会话上下文声明以根据计算资源限制访问权限。有关令牌中所包含声明的完整列表，请参阅了解令牌声明。

了解令牌声明

当您调用 GetWebIdentityToken API 时，Amazon Security Token Service 会返回一个已签名的 JSON Web 令牌（JWT），其中包含一组代表 IAM 主体身份的声明。这些令牌均符合 RFC 7519。了解这些令牌的结构和内容可帮助您实现安全的身份验证流程，在外部服务中配置适当的声明验证，并有效地使用自定义声明实现精细的访问控制。

JWT 包含标准 OpenID Connect（OIDC）声明，例如主题（“sub”）、受众（“aud”）、发布者（“iss”），以便于实现不同外部服务之间的互操作性。在适用的情况下，Amazon STS 会使用 Amazon 身份特定的声明（如 Amazon 账户 ID 和主体标签）和会话上下文声明（如 EC2 实例 ARN）填充令牌。您还可以通过将自定义声明作为请求标签传递给 GetWebIdentityToken API 将其添加到令牌。Amazon 身份特定的声明、会话上下文声明和自定义声明嵌套在令牌中的“https://sts.amazonaws.com/”命名空间下。

有关令牌中包含的声明列表，请参阅下面的示例令牌。请注意，所有这些声明可能不会同时出现在令牌中。

```
{
  "iss": "https://abc123-def456-ghi789-jkl012.tokens.sts.global.api.aws",
  "aud": "https://api.example.com",
  "sub": "arn:aws:iam::123456789012:role/DataProcessingRole",
  "iat": 1700000000,
  "exp": 1700000900,
  "jti": "xyz123-def456-ghi789-jkl012",
  "https://sts.amazonaws.com/": {
    "aws_account": "123456789012",
    "source_region": "us-east-1",
    "org_id": "o-abc1234567",
    "ou_path": "o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/",
    "principal_tags": {
      "environment": "production",
      "team": "data-engineering",
      "cost-center": "engineering"
    },
    "lambda_source_function_arn": "arn:aws:lambda:us-
east-1:123456789012:function:process-data",
    "request_tags": {
        "job-id": "job-2024-001",
        "priority": "high",
        "data-classification": "sensitive"
    }
  }
```

```
}
```

标准声明

令牌中存在的标准 OIDC 声明有助于实现与各种外部服务的互操作性。这些声明可以使用大多数 JWT 库进行验证。

声明	名称	描述	示例值
iss	发布者	账户特定的发布者 URL。外部服务会验证此声明，以确保其匹配可信发布者。	https://abc123-def456-ghi789-jkl012.tokens.sts.global.api.aws
aud	受众	GetWebIdentityToken 请求中指定的令牌的预期接收者。	https://api.example.com
sub	主题	请求令牌的 IAM 主体的 ARN。	arn:aws:iam::123456789012:role/DataProcessingRole
iat	发布时间	标识 JWT 发布时间的 NumericDate 值。	1,700,000,000
exp	过期	标识过期时间的 NumericDate 值，在该时间或之后不得接受 JWT 进行处理。	1,700,000,900
jti	JWT ID	此令牌实例的唯一标识符。	xyz123-def456-ghi789-jkl012

自定义声明

除了标准 OIDC 声明外，Amazon STS 还会添加关于身份和会话上下文的声明（如果适用）。您还可以通过将自有声明作为请求标签传递来将其添加到令牌。自定义声明嵌套在 https://sts.amazonaws.com/ 命名空间下。

Amazon 身份声明

这些声明提供有关您的 Amazon 账户、组织结构和 IAM 主体的详细信息。

声明	描述	条件键映射	示例值
aws_account	您的 Amazon 账户 ID	aws:PrincipalAccount	123456789012
source_region	请求令牌的 Amazon 区域	aws:RequestedRegion	us-east-1
org_id	您的 Amazon Organizations ID（如果您的账户是组织的成员）	aws:PrincipalOrgID	o-abc1234567
ou_path	您的组织单位路径（如果适用）	aws:PrincipalOrgPaths	o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
principal_tags	附加到 IAM 主体或代入角色会话的标签。请求令牌时，如果请求的 IAM 主体同时有主体标签和会话标签，则会话标签将出现在 JWT 中。	aws:PrincipalTag/<tag-key>	{"environment": "production", "team": "data-engineering", "cost-center":"engineering"}

会话上下文声明

这些声明提供有关发起令牌请求的计算环境和会话的信息。Amazon STS 会根据请求主体的会话上下文自动包含这些声明（如果适用）。

声明	描述	条件键映射	示例值
original_session_exp	原始角色会话凭证过期时间（适用于代入的角色）	不适用	2024-01-15T10:00:00Z
federated_provider	联合身份验证会话的身份提供商名称	aws:FederatedProvider	arn:aws:iam::11112

声明	描述	条件键映射	示例值
			2223333:oidc-provider/your_oidc_provider
identity_store_user_id	IAM Identity Center 用户 ID	identitystore:UserId	user-abc123def456
identity_store_arn	Identity Center 身份存储的 ARN	identitystore:IdentityStoreArn	arn:aws:identitystore::123456789012:identitystore/d-abc1234567
ec2_source_instance_arn	请求 EC2 实例的 ARN	ec2:SourceInstanceArn	arn:aws:ec2:us-east-1:123456789012:instance/i-abc123def456
ec2_instance_source_vpc	提供 EC2 角色凭证的 VPC ID	aws:Ec2InstanceSourceVpc	vpc-abc123def456
ec2_instance_source_private_ipv4	EC2 实例的私有 IPv4 地址	aws:Ec2InstanceSourcePrivateIPv4	10.0.1.25
ec2_role_delivery	实例元数据服务版本	ec2:RoleDelivery	2
source_identity	主体设置的来源身份	aws:SourceIdentity	admin-user
lambda_source_function_arn	调用 Lambda 函数的 ARN	lambda:SourceFunctionArn	arn:aws:lambda:us-east-1:123456789012:function:my-function
glue_credential_issuing_service	Glue 作业的 Amazon Glue 服务标识符	glue:CredentialIssuingService	glue.amazonaws.com

请求标签

您可以通过在 GetWebIdentityToken API 请求中指定标签将自定义声明添加到令牌。这些声明在令牌中的 request_tags 字段下显示，使您能够传递可供外部服务用于精细授权决策的特定信息。每个请求最多可以指定 50 个标签。

请求示例：

```
response = sts_client.get_web_identity_token(
    Audience=['https://api.example.com'],
    SigningAlgorithm='ES384',
    Tags=[
        {'Key': 'team', 'Value': 'data-engineering'},
        {'Key': 'cost-center', 'Value': 'analytics'},
        {'Key': 'environment', 'Value': 'production'}
    ]
)
```

令牌中由此产生的声明：

```
{
  "request_tags": {
    "team": "data-engineering",
    "cost-center": "analytics",
    "environment": "production"
  }
}
```

使用 IAM 策略控制访问权限

IAM 提供多种策略类型来控制对外站身份联合验证功能的访问权限。您可以使用基于身份的策略控制哪些 IAM 主体可以请求令牌并强制执行特定的令牌属性，例如受众、生命周期和签名算法。服务控制策略（SCP）使您能够对 Amazon Organizations 中所有账户的令牌生成强制执行组织范围的限制。资源控制策略（RCP）控制资源级别的访问权限。您还可以使用 VPC 端点策略限制哪些主体可以通过您的 VPC 端点访问 Amazon STS `GetWebIdentityToken` API，从而为您的安全状况添加网络级控制。本节说明如何使用这些策略类型和条件键实现精细的访问控制。

要请求身份令牌，IAM 主体必须拥有 `sts:GetWebIdentityToken` 权限。通过附加到 IAM 用户或角色的身份策略授予此权限。要允许将标签（键、值对）传递给 GetWebIdentityToken 调用，IAM 主体必须拥有 `sts:TagGetWebIdentityToken` 权限。

- 使用 sts:IdentityTokenAudience 条件键限制哪些外部服务可以接收令牌。
- 使用 sts:DurationSeconds 条件键强制执行最长令牌生命周期。
- 使用 sts:SigningAlgorithm 条件键要求特定的加密算法。
- 使用 aws:RequestTag 条件键可将请求中传递的标签键值对与您在策略中指定的标签对进行比较。
- 使用 aws:TagKeys 条件键可将请求中的标签键与您在策略中指定的键进行比较。

要详细了解 IAM 策略中提供的条件键，请参阅 IAM 和 Amazon STS 条件键。

此示例身份策略组合了多个条件键：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowTokenGenerationWithRestrictions",
            "Effect": "Allow",
            "Action": "sts:GetWebIdentityToken",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "sts:IdentityTokenAudience": [
                        "https://api1.example.com",
                        "https://api2.example.com"
                    ]
                },
                "NumericLessThanEquals": {
                    "sts:DurationSeconds": 300
                },
                "StringEquals": {
                    "sts:SigningAlgorithm": "ES384"
                }
            }
        }
    ]
}
```

最佳实践

请遵循以下建议，将您的 Amazon 身份安全地联合到外部服务。

- 使用较短的令牌生命周期：请求满足您运营需求的生命周期最短的令牌。

- 使用 IAM 策略实施最低权限访问并限制令牌属性：仅向需要 `sts:GetWebIdentityToken` 权限的 IAM 主体授予该权限。根据需要使用条件键指定签名算法、允许的令牌受众和最长令牌生命周期。
- 验证外部服务中的声明：为安全起见，请务必验证主题（“sub”）、受众（“aud”）等相关声明，以确保其匹配您的预期值。尽可能验证自定义声明，以便在外部服务中实现精细授权决策。

IAM 临时安全凭证

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给受信任用户。临时安全凭证的工作方式与长期访问密钥凭证的工作方式几乎相同，仅存在以下差异：

- 顾名思义，临时安全凭证是短期 凭证。可将这些凭证的有效时间配置几分钟到几小时。一旦这些凭证到期，Amazon 将不再识别这些凭证或不再允许来自使用这些凭证发出的 API 请求的任何类型的访问。
- 临时安全凭证不随用户一起存储，而是动态生成并在用户提出请求时提供给用户。临时安全凭证到期时 (甚至之前)，用户可以请求新的凭证，只要请求凭证的用户仍有权这样做。

因此，与长期凭证相比，临时凭证具有以下优势：

- 您不必随应用程序分配或嵌入长期 Amazon 安全凭证。
- 可允许用户访问您的 Amazon 资源，而不必为这些用户定义 Amazon 身份。临时凭证是角色和身份联合验证的基础。
- 临时安全凭证的生命周期较短，因此无需更新或在不再需要这些凭证时显式撤销这些凭证。临时安全凭证到期后无法重复使用。您可指定凭证的有效期，但有最长限期。

Amazon STS 和 Amazon 区域

临时安全凭证由 Amazon STS 生成。默认情况下，Amazon STS 是一种全球服务，在 `https://sts.amazonaws.com` 上具有单个终端节点。不过，您也可以选择对任何其他支持的区域中的终端节点进行 Amazon STS API 调用。这可能会将请求发送到地理位置离您较近的区域中的服务器以减少延迟（服务器延迟）。无论您的凭证来自于哪个区域，它们都会在全球范围内起作用。有关更多信息，请参阅 管理 Amazon Web Services 区域中的 Amazon STS。

临时凭证的常见情形

临时凭证在涉及联合身份验证、委派、跨账户访问和 IAM 角色的情形中很有用。

联合身份

您可以在 Amazon 之外的外部系统中管理用户身份，并为从这些系统登录的用户授予访问权限以执行 Amazon 任务和访问您的 Amazon 资源。IAM 支持两种联合身份验证。在两种情况下，身份都存储在 Amazon 外部。区别在于外部系统所处的位置 在您的数据中心内或 Web 上的外部第三方。要比较用于身份联合验证的临时安全凭证的功能，请参阅 比较 Amazon STS 凭证。

有关外部身份提供程序的更多信息，请参阅身份提供程序和 Amazon 中的联合身份验证。

- OpenID Connect（OIDC）联合身份验证：您可让用户使用已知的第三方身份提供商（例如，Login with Amazon、Facebook、Google 或任何 OIDC 兼容的提供商）登录。对于移动或 Web 应用程序，无需创建自定义登录代码或管理您自己的用户身份。使用 OIDC 联合身份验证可帮助您确保 Amazon Web Services 账户 的安全，因为您不必随应用程序分配长期安全凭证（如 IAM 用户访问密钥）。有关更多信息，请参阅 OIDC 联合身份验证。

  Amazon STS OIDC 联合身份验证支持 Login with Amazon、Facebook、Google 和任何 OpenID Connect（OIDC）兼容的身份提供者。

  > **Note**
  >
  > 有关移动应用程序，我们建议使用 Amazon Cognito。您可以将此服务与面向移动开发的 Amazon 开发工具包搭配使用，为用户创建唯一的身份，并对其进行身份验证，以使其安全地访问您的 Amazon 资源。Amazon Cognito 支持与 Amazon STS 相同的身份提供程序，还支持未经身份验证的（来宾）访问，并允许您在用户登录时迁移用户数据。Amazon Cognito 还提供用于同步用户数据的 API 操作，因此，无论用户在设备间怎样转移，其数据总能得以保留。有关更多信息，请参阅《Amplify 文档》中的使用 Amplify 进行身份验证。

- SAML 联合身份验证 – 您可以对您的企业网络中的用户进行身份验证，然后为这些用户提供对 Amazon 的访问权限，而无需为其创建新的 Amazon 身份，他们也无需使用不同的登录凭证进行登录。这称为用于临时访问的单点登录方法。Amazon STS 支持开放标准 [如安全断言标记语言（SAML）2.0]，这使您可以使用 Microsoft AD FS 来利用您的 Microsoft Active Directory。您还可以使用 SAML 2.0 管理自己的联合身份用户身份解决方案。有关更多信息，请参阅 SAML 2.0 联合身份验证。
  - 自定义联合身份代理 - 您可以使用您企业的身份验证系统来授予对 Amazon 资源的访问权限。有关示例方案，请参阅使自定义身份凭证代理程序能够访问 Amazon 控制台。
  - 使用 SAML 2.0 的联合 - 您可以使用您企业的身份验证系统和 SAML 来授予对 Amazon 资源的访问权限。有关更多信息以及示例方案，请参阅SAML 2.0 联合身份验证。

用于跨账户访问的角色

很多组织均维护多个 Amazon Web Services 账户。利用角色和跨账户访问，您可以在一个账户中定义用户身份，并使用这些身份访问属于您组织的其他账户中的 Amazon 资源。这称为用于临时访问的委派方法。有关创建跨账户角色的更多信息，请参阅 创建角色，向 IAM 用户授予权限。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

适用于 Amazon EC2 的 角色

如果您在 Amazon EC2 实例上运行一些应用程序，并且这些应用程序需要访问 Amazon 资源，则您可在启动您的实例时为其提供临时安全凭证。这些临时安全凭证对该实例上运行的所有应用程序都可用，因此您无需在该实例上存储任何长期凭证。有关更多信息，请参阅 使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限。

要了解有关 IAM Amazon EC2 角色凭证的更多信息，请参阅《Amazon Elastic Compute Cloud 用户指南》中的 Amazon EC2 的 IAM 角色。

其他 Amazon 服务

可使用临时安全凭证访问大多数 Amazon 服务。有关接受临时安全凭证的服务的列表，请参阅使用 IAM 的 Amazon 服务。

使用临时凭证的示例应用程序

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅 IAM 临时安全凭证。要了解如何使用 Amazon STS 管理临时安全凭证，可下载以下示例应用程序，其中实现完整的示例方案：

- 使用 Windows Active Directory、ADFS 和 SAML 2.0 启用 Amazon 的联合身份验证。演示如何使用 Windows Active Directory（AD）、Active Directory 联合身份验证服务（ADFS）2.0 和 SAML（安全断言标记语言）2.0，通过企业联合身份验证将访问权限委派给 Amazon。
- 使自定义身份凭证代理程序能够访问 Amazon 控制台. 说明如何创建启用单点登录（SSO）的自定义联合身份验证代理，以使现有 Active Directory 用户能够登录到 Amazon Web Services 管理控制台。
- 如何使用 Shibboleth 单点登录到 Amazon Web Services 管理控制台。介绍如何使用 Shibboleth 和 SAML 为用户提供对 Amazon Web Services 管理控制台的单点登录 (SSO) 访问权限。

OIDC 联合身份验证示例

以下示例应用程序说明如何使用 Login with Amazon、Amazon Cognito、Facebook 或 Google 等提供者的 OIDC 联合身份验证。您可以将从这些提供商处获得的身份验证信息作为临时 Amazon 安全凭证以访问 Amazon 服务。

- Amazon Cognito 教程 — 建议您将 Amazon Cognito 与适用于移动开发的 Amazon SDK 搭配使用。Amazon Cognito 是管理适用于移动应用程序的身份的最简单方式，它提供了同步和跨设备身份等额外功能。有关 Amazon Cognito 的更多信息，请参阅《Amplify 文档》中的使用 Amplify 进行身份验证。

有关临时安全凭证的其他资源

以下场景和应用可指导您使用临时安全凭证：

- 如何将 Amazon STS SourceIdentity 与您的身份提供者集成。本文向您介绍当使用 Okta、Ping 或 OneLogin 作为 IdP 时如何设置 Amazon STS `SourceIdentity` 属性。
- OIDC 联合身份验证。本节讨论如何在使用 OIDC 联合身份验证和 `AssumeRoleWithWebIdentity` API 时配置 IAM 角色。
- 使用 MFA 保护 API 访问. 本主题介绍如何使用角色要求多重身份验证 (MFA)，以保护您账户中的敏感 API 操作。

有关 Amazon 中的策略和权限的更多信息，请参阅以下主题：

- 适用于 Amazon 资源的 Access Management
- 策略评估逻辑.
- Amazon Simple Storage Service 用户指南中的管理对 Amazon S3 资源的访问权限。
- 要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

比较 Amazon STS 凭证

下表比较了 Amazon STS 中返回临时安全凭证的 API 操作的功能。要了解在担任角色以请求临时安全凭证时使用的各种方法，请参阅担任角色的方法。要了解允许您传递会话标签的不同 Amazon STS API 操作，请参阅在 Amazon STS 中传递会话标签。

> **Note**
>
> 您可以将 Amazon STS API 调用发送到全球终端节点或某个区域终端节点。如果您选择更靠近您的终端节点，则可减少延迟并改善 API 调用的性能。如果您不再能与原始终端节点进行通信，也可选择将调用定向到替代的区域终端节点。如果您使用某种 Amazon SDK，请先使用该开发工具包的方法指定一个区域，然后再进行 API 调用。如果您正在手动构建 HTTP API 请求，则必须自行将请求定向到正确的终端节点。有关更多信息，请参阅区域和终端节点和 管理 Amazon Web Services 区域中的 Amazon STS 的 Amazon STS 部分。

Amazon STS API	谁能调用	凭证生命周期 (最小值 | 最大值 | 默认值)	MFA 支持[1]	会话策略支持[2]	对生成的临时凭证的限制
AssumeRe	具有现有临时安全凭证的 IAM 用户或 IAM 角色	15 分钟 | 最大会话持续时间设置[3] | 1 小时	支持	是	无法调用 GetFederationToken 或 GetSessionToken 。
AssumeReWithSAM	任何用户；发起人必须传递 SAML 身份验证响应，指示身份验证来自已知的身份提供程序	15 分钟 | 最大会话持续时间设置[3] | 1 小时	否	是	无法调用 GetFederationToken 或 GetSessionToken 。
AssumeReWithWebIdentity	任何用户；发起人必须传递 OIDC 兼容的 JWT 令牌，指示身份验证来自已知的身份提供者	15 分钟 | 最大会话持续时间设置[3] | 1 小时	否	是	无法调用 GetFederationToken 或 GetSessionToken 。

Amazon STS API	谁能调用	凭证生命周期 (最小值 | 最大值 | 默认值)	MFA 支持[1]	会话策略支持[2]	对生成的临时凭证的限制
GetFederationToken	IAM 用户或 Amazon Web Services 账户根用户	IAM 用户：15 分钟 | 36 小时 | 12 小时 根用户：15 分钟 | 1 小时 | 1 小时	否	是	无法使用 Amazon CLI 或 Amazon API 调用 IAM 操作。此限制不适用于控制台会话。 无法调用除 `GetCallerIdentity` 之外的 Amazon STS 操作。[4] 允许通过 SSO 登录到控制台。[5]
GetSessionToken	IAM 用户或 Amazon Web Services 账户根用户	IAM 用户：15 分钟 | 36 小时 | 12 小时 根用户：15 分钟 | 1 小时 | 1 小时	是	否	除非请求附带了 MFA 信息，否则无法调用 IAM API 操作。 无法调用除 `AssumeRole` 或 `GetCallerIdentity` 之外的 Amazon STS API 操作。 不允许通过 SSO 登录到控制台。[6]

[1] MFA 支持。在调用 AssumeRole 和 GetSessionToken API 操作时，可以包含有关 Multi-Factor Authentication (MFA) 设备的信息。这可确保通过 API 调用生成的临时安全凭证只能由已使用 MFA 设备进行身份验证的用户使用。有关更多信息，请参阅 使用 MFA 保护 API 访问。

[2] 会话策略支持。会话策略是当您以编程方式为角色或 Amazon STS 联合用户会话创建临时会话时作为参数传递的策略。此策略限制来自角色或用户的基于身份的策略且分配给该会话的权限。生成的会话的权限是实体的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅会话策略。

[3] 最大会话持续时间设置。可以使用 `DurationSeconds` 参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。

[4] GetCallerIdentity。无需权限即可执行该操作。如果管理员为您的 IAM 用户或角色添加明确拒绝访问 `sts:GetCallerIdentity` 操作的策略，您仍然可以执行该操作。不需要权限，因为在拒绝 IAM 用户或角色访问时返回相同的信息。要查看示例响应，请参阅我没有权限执行：iam:DeleteVirtualMFADevice。

[5] 通过单一登录 (SSO) 登录到控制台。为了支持 SSO，Amazon 可让您调用联合终端节点 (`https://signin.aws.amazon.com/federation`) 并传递临时安全凭证。终端节点将返回一个令牌，可使用该令牌构建一个让用户直接登录到控制台而无需使用密码的 URL。有关更多信息，请参阅使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台 安全性博客中的和Amazon如何启用对 管理控制台的跨账户访问Amazon。

[6] 在检索您的临时凭证后，无法将该凭证传递到联合单一登录终端节点以访问 Amazon Web Services 管理控制台。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。

服务持有者令牌

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。这些服务支持一种协议，该协议要求您使用持有者令牌，而不是使用传统的 适用于 API 请求的 Amazon 签名版本 4。当您执行需要持有者令牌的 Amazon CLI 或 Amazon API 操作时，Amazon 服务将代表您请求持有者令牌。该服务将为您提供令牌，然后您可以使用该令牌在该服务中执行后续操作。

Amazon STS 服务持有者令牌包括来自您的原始主体身份验证的信息，这些信息可能会影响您的权限。这些信息可以包括主体标签、会话标签和会话策略。令牌的访问密钥 ID 以 `ABIA` 前缀开头。这可帮助您在 CloudTrail 日志中识别使用服务持有者令牌执行的操作。

> **Important**
>
> 持有者令牌只能用于对生成它的服务的调用以及生成它的区域。您不能使用持有者令牌在其他服务或区域中执行操作。

支持持有者令牌的服务示例是 Amazon CodeArtifact。在使用 NPM、Maven 或 PIP 等程序包管理器与 Amazon CodeArtifact 进行交互之前，您必须调用 `aws codeartifact get-authorization-token` 操作。此操作返回可用于执行 Amazon CodeArtifact 操作的持有者令牌。或者，您可以使用 `aws codeartifact login` 命令完成相同的操作，然后自动配置您的客户端。

如果您在为您生成持有者令牌的 Amazon 服务中执行操作，则您的 IAM policy 中必须具有以下权限：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServiceBearerToken",
            "Effect": "Allow",
            "Action": "sts:GetServiceBearerToken",
            "Resource": "*"
        }
    ]
}
```

有关服务持有者令牌的示例，请参阅《Amazon CodeArtifact 用户指南》中的为 Amazon CodeArtifact 使用基于身份的策略。

请求临时安全凭证

要请求临时安全凭证，您可以在 Amazon API 中使用 Amazon Security Token Service (Amazon STS) 操作。这些操作包括创建受信任用户，并为其提供可以控制 Amazon 资源访问的临时安全凭证。有关 Amazon STS 的更多信息，请参阅 IAM 临时安全凭证。要了解在担任角色以请求临时安全凭证时使用的各种方法，请参阅担任角色的方法。

要调用 API 操作，您可以使用其中的一个 Amazon SDK。这些开发工具包适用于各种不同的编程语言和环境，包括 Java、.NET、Python、Ruby、Android 和 iOS。这些开发工具包负责处理各种任务，如以加密方式对您的请求进行签名、在必要时重试请求以及处理错误响应。还可使用 Amazon STS 查询 API（在 Amazon Security Token Service API 参考中介绍）。最后，两个命令行工具支持 Amazon STS 命令：Amazon Command Line Interface和Amazon Tools for Windows PowerShell。

Amazon STS API 操作使用临时安全凭证（包括访问密钥对和会话令牌）创建新会话。访问密钥对由访问密钥 ID 和私有密钥组成。用户 (或用户所运行的应用程序) 可使用这些凭证访问您的资源。您可以

使用 Amazon STS API 操作以编程方式创建角色会话和传递会话策略及会话标签。生成的会话的权限是角色的基于身份的策略与会话策略的交集。有关会话策略的更多信息，请参阅会话策略。有关会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。

> **Note**
>
> Amazon STS API 操作返回的会话令牌大小不固定。我们强烈建议不要假设最大大小。典型的令牌大小小于 4096 字节，但可能会发生变化。

通过 Amazon 区域使用 Amazon STS

您可以将 Amazon STS API 调用发送到全球终端节点或某个区域终端节点。如果您选择更靠近您的终端节点，则可减少延迟并改善 API 调用的性能。如果您不再能与原始终端节点进行通信，也可选择将调用定向到替代的区域终端节点。如果您使用某种 Amazon SDK，请先使用该开发工具包的方法指定一个区域，然后再进行 API 调用。如果您正在手动构建 HTTP API 请求，则必须自行将请求定向到正确的终端节点。有关更多信息，请参阅区域和终端节点和 管理 Amazon Web Services 区域中的 Amazon STS 的 Amazon STS 部分。

下面是让您可以用来获取临时凭证以用于您的 Amazon 环境和应用程序的 API 操作。

请求通过自定义身份凭证代理程序进行跨账户委派和联合身份验证的凭证

`AssumeRole` API 操作对于允许现有 IAM 用户访问其没有访问权限的 Amazon 资源很有用。例如，用户可能需要对其他 Amazon Web Services 账户 中资源的访问权限。此外，可使用它来暂时获得特权访问权限 - 例如，提供多重身份验证 (MFA)。您必须使用活动凭证调用该 API。要了解谁可以调用此操作，请参阅 比较 Amazon STS 凭证。有关更多信息，请参阅创建角色，向 IAM 用户授予权限和使用 MFA 保护 API 访问。

要请求通过自定义身份凭证代理程序进行跨账户委派和联合身份验证的临时安全凭证

1. 使用您的 Amazon 安全凭证进行身份验证。必须使用有效的 Amazon 安全凭证来进行此调用。
2. 调用 `AssumeRole` 操作。

以下示例显示了使用 `AssumeRole` 的示例请求和响应。此示例请求将在指定的持续时间内担任 `demo` 角色，其中包含会话策略、会话标签、外部 ID 和源身份。生成的会话命名为 `John-session`。

Example示例请求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=AssumeRole
&RoleSessionName=John-session
&RoleArn=arn:aws:iam::123456789012:role/demo
&Policy=%7B%22Version%22%3A%222012-10-17       %22%2C%22Statement%22%3A%5B%7B%22Sid
%22%3A%20%22Stmt1%22%2C%22Effect%22%3A%20%22Allow%22%2C%22Action%22%3A%20%22s3%3A*
%22%2C%22Resource%22%3A%20%22*%22%7D%5D%7D
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&ExternalId=123ABC
&SourceIdentity=DevUser123
&AUTHPARAMS
```

在前面的示例中显示的策略值是以下策略的 URL 编码版本：

JSON

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"Stmt1","Effect":"Allow","Action":"s3:*","Resource":"*"}]}
```

示例中的 AUTHPARAMS 参数是您的签名 的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 Amazon 开发工具包创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请参阅《Amazon Web Services 一般参考》中的 使用签名版本 4 签署 Amazon 请求，以了解如何签署请求。

除了临时安全凭证之外，该响应还包括联合身份用户的 Amazon Resource Name (ARN) 和凭证的到期时间。

Example响应示例

```
<AssumeRoleResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<AssumeRoleResult>
<SourceIdentity>DevUser123</SourceIdentity>
<Credentials>
```

```
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
   9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
   +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCR/oLxBA==
  </SessionToken>
  <SecretAccessKey>
   wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-07-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
<AssumedRoleUser>
  <Arn>arn:aws:sts::123456789012:assumed-role/demo/John</Arn>
  <AssumedRoleId>ARO123EXAMPLE123:John</AssumedRoleId>
</AssumedRoleUser>
<PackedPolicySize>8</PackedPolicySize>
</AssumeRoleResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</AssumeRoleResponse>
```

> **Note**
>
> Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。即使您的纯文本符合其他要求，您的请求也可能会由于此限制而失败。`PackedPolicySize` 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。

通过 OIDC 提供商请求凭证

`AssumeRoleWithWebIdentity` API 操作返回一组临时的 Amazon 安全凭证，以换取 JSON Web 令牌（JWT）。这包括公共身份提供者，例如，使用 Login with Amazon、Facebook、Google，以及发布与 OpenID Connect（OIDC）发现兼容的 JWT 的提供者，例如 GitHub 操作或 Azure Devops。有关更多信息，请参阅 OIDC 联合身份验证。

> **Note**
>
> `AssumeRoleWithWebIdentity` 请求未使用签名，也不需要 Amazon 凭证。

通过 OIDC 提供商请求凭证

1. 调用 AssumeRoleWithWebIdentity 操作。

   当您调用 AssumeRoleWithWebIdentity 时，Amazon 会使用通过 IdP 的 JSON Web 密钥集（JWKS）提供的公钥验证数字签名，从而验证所提供的令牌。如果令牌有效，并且满足 IAM 角色信任策略中规定的所有条件，则 Amazon 会向您返回以下信息：

   - 一组临时安全凭证。这些证书由访问密钥 ID、秘密访问密钥和会话令牌组成。
   - 所担任角色的角色 ID 和 ARN。
   - 一个 SubjectFromWebIdentityToken 值，其中包含独一无二的用户 ID。
2. 然后，您的应用程序可以使用响应中返回的临时安全凭证进行 Amazon API 调用。该过程与使用长期安全凭证进行 Amazon API 调用相同。不同之处在于，您必须包含会话令牌，以便 Amazon 验证临时安全凭证是否有效。

您的应用程序应缓存 Amazon STS 返回的凭证，并根据需要刷新凭证。如果您的应用程序是使用 Amazon SDK 构建的，则 SDK 的凭证提供者可以在凭证过期之前处理调用 AssumeRoleWithWebIdentity 和刷新 Amazon 凭证。有关更多信息，请参阅《Amazon SDK 和工具参考指南》中的《Amazon SDK 和工具标准化凭证提供者》https://docs.amazonaws.cn/sdkref/latest/guide/standardized-credentials.html。

通过 SAML 2.0 身份提供商请求凭证

AssumeRoleWithSAML API 操作为通过您的组织的现有身份系统验证的 SAML 联合主体返回一组临时安全凭证。用户还必须使用 SAML 2.0（安全断言标记语言）将身份验证和授权信息传递给 Amazon。对于将身份系统 (如 Windows Active Directory 或 OpenLDAP) 与可生成 SAML 断言的软件集成在一起的组织，该 API 操作是非常有用的。此类集成提供有关用户身份和权限 (如 Active Directory 联合身份验证服务或 Shibboleth) 的信息。有关更多信息，请参阅 SAML 2.0 联合身份验证。

1. 调用 AssumeRoleWithSAML 操作。

   这是一个未签名的调用，意味着您无需在发出请求之前对 Amazon 安全凭证进行身份验证。

   > **Note**
   >
   > 对 AssumeRoleWithSAML 的调用未签名 (加密)。因此，只有在通过受信任的中介传输请求时，才应包含可选的会话策略。在这种情况下，有人可能会修改策略以删除限制。

2. 调用 AssumeRoleWithSAML 时，Amazon 将验证 SAML 断言的真实性。假定身份提供程序证实断言有效，则 Amazon 向您返回以下信息：

   - 一组临时安全凭证。这些证书由访问密钥 ID、秘密访问密钥和会话令牌组成。
   - 所担任角色的角色 ID 和 ARN。
   - Audience 值，包含 SAML 断言的 Recipient 元素的 SubjectConfirmationData 属性值。
   - Issuer 值，包含 SAML 断言的 Issuer 元素值。
   - NameQualifier 元素，包含通过 Issuer 值、Amazon Web Services 账户 ID 以及 SAML 提供商的易记名称生成的哈希值。与 Subject 元素相结合时，它们可以唯一地标识 SAML 联合主体。
   - Subject 元素，包含 SAML 断言的 NameID 元素的 Subject 元素值。
   - SubjectType 元素，指示 Subject 元素的格式。值可以是 persistent、transient 或在您的 SAML 断言中使用的 Format 和 Subject 元素的完整 NameID URI。有关 NameID 元素的 Format 属性的信息，请参阅为身份验证响应配置 SAML 断言。。

3. 使用响应中返回的临时安全凭证进行 Amazon API 调用。该过程与使用长期安全凭证进行 Amazon API 调用相同。不同之处在于，您必须包含会话令牌，以便 Amazon 验证临时安全凭证是否有效。

您的应用程序应缓存凭证。默认情况下，凭证在 1 小时后到期。如果未使用 软件开发工具包中的 AmazonSTSCredentialsProviderAmazon 操作，则由您和您的应用程序负责再次调用 AssumeRoleWithSAML。在旧安全凭证到期之前，可以调用该操作以获取一组新的临时安全凭证。

通过自定义身份凭证代理程序请求凭证

GetFederationToken API 操作为 Amazon STS 联合用户主体返回一组临时安全凭证。该 API 不同于 AssumeRole，其默认有效期大大增加 (12 小时而不是 1 小时)。此外，您还可以使用 DurationSeconds 参数指定临时安全凭证保持有效的持续时间。生成的凭证在指定的时间段内有效，介于 900 秒（15 分钟）到 129600 秒（36 小时）之间。较长的有效期有助于减少调用 Amazon 的次数，因为您不需要经常获取新凭证。

1. 使用您的特定 IAM 用户的 Amazon 安全凭证进行身份验证。必须使用有效的 Amazon 安全凭证来进行此调用。
2. 调用 GetFederationToken 操作。

GetFederationToken 调用将返回临时安全凭证，其中包括会话令牌、访问密钥、私有密钥和到期时间。如果要在组织内管理权限 (例如，使用代理应用程序分配权限)，则可使用 GetFederationToken。

以下示例显示了使用 GetFederationToken 的示例请求和响应。此示例请求使用会话策略 ARN 和会话标签，在指定的持续时间内联合调用用户的身份。生成的会话命名为 Jane-session。

Example示例请求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetFederationToken
&Name=Jane-session
&PolicyArns.member.1.arn==arn%3Aaws%3Aiam%3A%3A123456789012%3Apolicy%2FRole1policy
&DurationSeconds=1800
&Tags.member.1.Key=Project
&Tags.member.1.Value=Pegasus
&Tags.member.2.Key=Cost-Center
&Tags.member.2.Value=12345
&AUTHPARAMS
```

上述示例中显示的策略 ARN 包含以下 URL 编码的 ARN：

```
arn:aws:iam::123456789012:policy/Role1policy
```

还要注意，该示例中的 &AUTHPARAMS 参数用作身份验证信息的占位符。这是签名，您必须将其包括在 Amazon HTTP API 请求中。建议使用 Amazon 开发工具包创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请访问《Amazon Web Services 一般参考》中的 使用签名版本 4 签署 Amazon 请求，以了解如何签署请求。

除了临时安全凭证之外，该响应还包括联合身份用户的 Amazon Resource Name (ARN) 和凭证的到期时间。

Example响应示例

```
<GetFederationTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetFederationTokenResult>
<Credentials>
  <SessionToken>
   AQoDYXdzEPT//////////wEXAMPLEtc764bNrC9SAPBSM22wDOk4x4HIZ8j4FZTwdQW
   LWsKWHGBuFqwAeMicRXmxfpSPfIeoIYRqTflfKD8YUuwthAx7mSEI/qkPpKPi/kMcGd
   QrmGdeehM4IC1NtBmUpp2wUE8phUZampKsburEDy0KPkyQDYwT7WZ0wq5VSXDvp75YU
```

```
    9HFvlRd8Tx6q6fE8YQcHNVXAkiY9q6d+xo0rKwT38xVqr7ZD0u0iPPkUL64lIZbqBAz
    +scqKmlzm8FDrypNC9Yjc8fPOLn9FX9KSYvKTr4rvx3iSIlTJabIQwj2ICCEXAMPLE==
  </SessionToken>
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2019-04-15T23:28:33.359Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE;</AccessKeyId>
</Credentials>
<FederatedUser>
  <Arn>arn:aws:sts::123456789012:federated-user/Jean</Arn>
  <FederatedUserId>123456789012:Jean</FederatedUserId>
</FederatedUser>
<PackedPolicySize>4</PackedPolicySize>
</GetFederationTokenResult>
<ResponseMetadata>
<RequestId>c6104cbe-af31-11e0-8154-cbc7ccf896c7</RequestId>
</ResponseMetadata>
</GetFederationTokenResponse>
```

> **Note**
>
> Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。即使您的纯文本符合其他要求，您的请求也可能会由于此限制而失败。`PackedPolicySize` 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。

Amazon 建议您在资源级别授予权限（例如，将基于资源的策略附加到 Amazon S3 存储桶），则可省略 `Policy` 参数。但是，如果不加入针对 Amazon STS 联合用户主体的策略，则临时安全凭证将不授予任何权限。在这种情况下，您必须 使用资源策略为联合身份用户授予您的 Amazon 资源的访问权限。

例如，假设您的 Amazon Web Services 账户 号码为 111122223333，且您拥有希望允许 Susan 访问的 Amazon S3 存储桶。Susan 的临时安全凭证不包括该存储桶的策略。在这种情况下，您需要确保该存储桶的策略具有与 Susan 的 ARN (如 `arn:aws:sts::111122223333:federated-user/Susan`) 匹配的 ARN。

为不受信任的环境中的用户请求凭证

`GetSessionToken` API 操作向现有 IAM 用户返回一组临时安全凭证。此 API 对提高安全性非常有用，例如在已针对 IAM 用户启用 MFA 的情况下提出 Amazon 请求。由于凭证是临时的，因此，在

IAM 用户通过不太安全的环境访问您的资源时，它们提供了增强的安全性。不太安全的环境示例包括移动设备或 Web 浏览器。

1. 使用您的特定 IAM 用户的 Amazon 安全凭证进行身份验证。必须使用有效的 Amazon 安全凭证来进行此调用。
2. 调用 `GetSessionToken` 操作。
3. `GetSessionToken` 将返回临时安全凭证，其中包括会话令牌、访问密钥 ID 和秘密访问密钥。

默认情况下，IAM 用户的临时安全凭证的有效期最长为 12 小时。但是，您可以使用 `DurationSeconds` 参数请求短至 15 分钟或长达 36 小时的持续时间。出于安全考虑，Amazon Web Services 账户根用户的令牌有效期仅为一小时。

以下示例显示了使用 `GetSessionToken` 的示例请求和响应。响应也包括了临时安全凭证的过期时间。

Example示例请求

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=1800
&AUTHPARAMS
```

示例中的 AUTHPARAMS 参数是您的签名 的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 Amazon 开发工具包创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请访问《Amazon Web Services 一般参考》中的 使用签名版本 4 签署 Amazon 请求，以了解如何签署请求。

Example响应示例

```
<GetSessionTokenResponse xmlns="https://sts.amazonaws.com/doc/2011-06-15/">
<GetSessionTokenResult>
<Credentials>
  <SessionToken>
   AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT+FvwqnKwRcOIfrRh3c/L
   To6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/IvU1dYUg2RVAJBanLiHb4IgRmpRV3z
   rkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/AXlzBBko7b15fjrBs2+cTQtp
   Z3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE
  </SessionToken>
```

```
  <SecretAccessKey>
  wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY
  </SecretAccessKey>
  <Expiration>2011-07-11T19:55:29.611Z</Expiration>
  <AccessKeyId>AKIAIOSFODNN7EXAMPLE</AccessKeyId>
</Credentials>
</GetSessionTokenResult>
<ResponseMetadata>
<RequestId>58c5dbae-abef-11e0-8cfe-09039844ac7d</RequestId>
</ResponseMetadata>
</GetSessionTokenResponse>
```

（可选）GetSessionToken 请求可能包括 SerialNumber 和 TokenCode 值以进行 Amazon Multi-Factor Authentication (MFA) 验证。如果提供的值有效，Amazon STS 将提供包含 MFA 身份验证状态的临时安全凭证。然后，可以使用临时安全凭证访问受 MFA 保护的 API 操作或 Amazon 网站，但前提是 MFA 身份验证有效。

下例展示一个 GetSessionToken 请求，其中包括 MFA 验证代码和设备序列号。

```
https://sts.amazonaws.com/
?Version=2011-06-15
&Action=GetSessionToken
&DurationSeconds=7200
&SerialNumber=YourMFADeviceSerialNumber
&TokenCode=123456
&AUTHPARAMS
```

> **Note**
>
> 对 Amazon STS 的调用可能针对的是全局端点，也可能针对的是激活您的 Amazon Web Services 账户 的任何区域终端节点。有关更多信息，请参阅区域和终端节点的 Amazon STS 部分。
>
> 示例中的 AUTHPARAMS 参数是您的签名 的占位符。签名是您必须在 Amazon HTTP API 请求中包含的身份验证信息。建议使用 Amazon 开发工具包创建 API 请求，这样做的一个好处是开发工具包将为您处理请求签名。如果您必须手动创建并签署 API 请求，请参阅《Amazon Web Services 一般参考》中的 使用签名版本 4 签署 Amazon 请求，以了解如何签署请求。

将临时凭证用于 Amazon 资源

借助 Amazon CLI 或 Amazon API（使用 Amazon SDK），您可以使用临时安全凭证提出对 Amazon 资源的编程请求。临时凭证提供的权限与长期安全凭证（例如 IAM 用户凭证）相同。但还是有几个区别：

- 使用临时安全凭证进行调用时，必须在调用中包含随这些临时凭证一同返回的会话令牌。Amazon 使用该会话令牌来验证临时安全凭证的有效性。
- 临时凭证在指定间隔后到期。临时凭证到期后，任何使用这些凭证进行的调用都将失败，因此您必须生成一组新的临时凭证。临时凭证的延期或刷新时间不得超过最初指定的时间间隔。
- 当您使用临时凭证发出请求时，您的主体可能会包含一组标签。这些标签来自附加到您代入的角色的会话标签和标签。有关会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。

如果您使用的是 Amazon SDK、Amazon Command Line Interface (Amazon CLI) 或 Tools for Windows PowerShell，则获取和使用临时安全凭证的方式随上下文而不同。如果您在 Amazon CLI 内部运行代码、 或 Tools for Windows PowerShell 命令，则可使用 Amazon EC2 的角色。否则，您可调用 Amazon STS API 来获取临时凭证，然后使用这些凭证显式地调用 Amazon 服务。

> **Note**
>
> 您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅IAM 临时安全凭证。Amazon STS 是一种全球服务，在 `https://sts.amazonaws.com` 上具有默认终端节点。此端点处于美国东部（弗吉尼亚州北部）区域，不过，您从此端点和其他端点获取的凭证在全球范围都有效。这些凭证可用于任何区域中的服务和资源。您也可以选择对任何支持的区域中的终端节点进行 Amazon STS API 调用。这可能会从地理位置离您较近的区域的服务器中发出请求以减少延迟。无论您的凭证来自于哪个区域，它们都会在全球范围内起作用。有关更多信息，请参阅 管理 Amazon Web Services 区域中的 Amazon STS。

目录

- 在 Amazon EC2 实例中使用临时凭证
- 将临时安全凭证用于 Amazon SDK。
- 将临时安全凭证用于 Amazon CLI。
- 将临时安全凭证用于 API 操作
- 更多信息

在 Amazon EC2 实例中使用临时凭证

如果要在 EC2 实例内部运行 Amazon CLI 命令或代码，建议的获取凭证的方法是使用 Amazon EC2 的角色。您可创建一个 IAM 角色，通过该角色指定要授予在 EC2 实例中运行的应用程序的权限。启动实例时，将该角色关联至实例。

之后，在该实例上运行的应用程序、Amazon CLI 和 Tools for Windows PowerShell 命令将能够从该实例的元数据获取自动临时安全凭证。您无需明确获取临时安全凭证。Amazon SDK、Amazon CLI 和 Tools for Windows PowerShell 自动从 EC2 实例元数据服务 (IMDS) 获取凭证并使用它们。临时凭证具有您为与该实例关联的角色定义的权限。

有关更多信息及示例，请参阅：

- 使用 IAM 角色授予对 Amazon Elastic Compute Cloud 上 Amazon 资源的访问权 — 适用于 Java 的 Amazon SDK
- 使用 IAM 角色授予访问权限 IAM - 适用于 .NET 的 Amazon SDK
- 创建角色 - 适用于 Ruby 的 Amazon SDK

将临时安全凭证用于 Amazon SDK。

要在代码中使用临时安全凭证，可以编程方式调用类似于 Amazon STS 的 `AssumeRole` API，提取生成的凭证和会话令牌。然后，您可以使用这些值作为对 Amazon 的后续调用的凭证。以下示例说明了有关使用 Amazon 开发工具包时如何使用临时安全凭证的伪代码：

```
assumeRoleResult = AssumeRole(role-arn);
tempCredentials = new SessionAWSCredentials(
   assumeRoleResult.AccessKeyId,
   assumeRoleResult.SecretAccessKey,
   assumeRoleResult.SessionToken);
s3Request = CreateAmazonS3Client(tempCredentials);
```

有关用 Python 编写的示例（使用 Amazon SDK for Python (Boto)），请参阅切换到 IAM 角色（Amazon API）。此示例说明如何调用 `AssumeRole` 以获取临时安全凭证，然后使用这些凭证调用 Amazon S3。

有关如何调用 `AssumeRole`、`GetFederationToken` 和其他 API 操作的详细信息，请参阅 Amazon Security Token Service API 参考。有关从结果中获取临时安全凭证和会话令牌的信息，请参阅所用开发工具包的文档。可以在主 Amazon 文档页上的 SDKs and Toolkits（开发工具包和工具箱）部分中找到所有 Amazon SDK 的文档。

您必须确保在旧凭证到期之前，获得一组新的凭证。在某些开发工具包中，可让提供商为您管理刷新凭证的过程；并可检查所用开发工具包的文档。

将临时安全凭证用于 Amazon CLI。

可将临时安全凭证用于 Amazon CLI。这对于测试策略来说很有用。

借助 Amazon CLI，您可以调用 `AssumeRole` 或 `GetFederationToken` 之类的 Amazon STS API，然后捕获输出结果。下面的示例说明了一个将输出发送至文件的 `AssumeRole` 调用。在示例中，假定 `profile` 参数是配置文件中的 Amazon CLI 配置文件。它还假定为有权代入角色的 IAM 用户引用凭证。

```
aws sts assume-role --role-arn arn:aws:iam::123456789012:role/role-name --role-session-
name "RoleSession1" --profile IAM-user-name > assume-role-output.txt
```

当命令完成后，您可以从路由到的任意位置中提取访问密钥 ID、秘密访问密钥和会话令牌。您可以手动或使用脚本执行此操作。之后，您可以将这些值分配给环境变量。

在运行 Amazon CLI 命令时，Amazon CLI 依特定顺序查找凭证 - 首先查找环境变量，然后是配置文件。因此，在将临时凭证放入环境变量后，Amazon CLI 会默认使用这些凭证。（如果在该命令中指定了 `profile` 参数，则 Amazon CLI 将跳过环境变量，而 Amazon CLI 在配置文件中查找，这使您能够根据需要覆盖环境变量中的凭证。）

下面的示例说明了如何为临时安全凭证设置环境变量然后调用 Amazon CLI 命令。由于未在 Amazon CLI 命令中包含 `profile` 参数，Amazon CLI 首先在环境变量中查找凭证，因而将使用该临时凭证。

Linux

```
$ export AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
$ export AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
$ export AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of session token>
$ aws ec2 describe-instances --region us-west-1
```

Windows

```
C:\> SET AWS_ACCESS_KEY_ID=ASIAIOSFODNN7EXAMPLE
C:\> SET AWS_SECRET_ACCESS_KEY=wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
C:\> SET AWS_SESSION_TOKEN=AQoDYXdzEJr...<remainder of token>
C:\> aws ec2 describe-instances --region us-west-1
```

将临时安全凭证用于 API 操作

如果是直接向 Amazon 发出 HTTPS API 请求，则可以使用从 Amazon Security Token Service (Amazon STS) 获取的临时安全凭证对这些请求进行签名。为此，您可以使用接收自 Amazon STS 的访问密钥 ID 和秘密访问密钥。您使用访问密钥 ID 和秘密访问密钥的方式，与使用长期凭证对请求签名是一样的。此外，还要将您从 Amazon STS 获得的会话令牌添加到 API 请求中。将会话令牌添加到 HTTP 标头或名为 `X-Amz-Security-Token` 的查询字符串参数中。将会话令牌添加到 HTTP 标头或 查询字符串参数，但不是同时添加到这两者。有关签署 HTTPS API 请求的更多信息，请参阅《Amazon Web Services 一般参考》中的 签署 Amazon API 请求。

更多信息

有关将 Amazon STS 与其他 Amazon 服务结合使用的更多信息，请参阅以下链接：

- Amazon S3。请参阅《Amazon Simple Storage Service 用户指南》中的 使用 IAM 用户临时凭证创建请求 或者 使用联合用户临时凭证创建请求。
- Amazon SNS。请参阅《Amazon Simple Notification Service 开发人员指南》中的将基于身份的策略用于 Amazon SNS。
- Amazon SQS。请参阅《Amazon Simple Queue Service 开发人员指南》中的 Amazon SQS 中的 Identity and Access Management。
- Amazon SimpleDB。请参阅 Amazon SimpleDB 开发人员指南中的使用临时安全凭证。

临时安全凭证的权限

您可以使用 Amazon Security Token Service (Amazon STS) 创建可控制对您的 Amazon 资源的访问的临时安全凭证，并将这些凭证提供给可信用户。有关 Amazon STS 的更多信息，请参阅 IAM 临时安全凭证。Amazon STS 颁发临时安全凭证后，这些凭证在到期时间之前有效，并且无法撤消。但是，由于每次使用证书发出请求时都会评估分配给临时安全凭证的权限，因此，即使证书已经签发，您仍可通过更改证书的访问权限来实现撤消证书的效果。

以下主题假定您具备一些 Amazon 权限和策略方面的知识。有关这些主题的更多信息，请参阅适用于 Amazon 资源的 Access Management。

主题

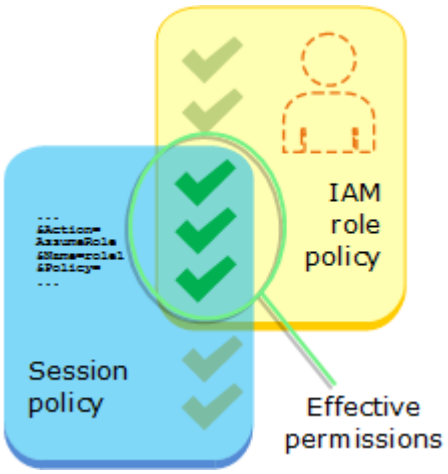
- AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 的权限
- 监控和控制使用所担任角色执行的操作

- GetFederationToken 的权限
- GetSessionToken 的权限
- 禁用临时安全凭证的权限
- 授予创建临时安全凭证的权限
- 授予使用身份增强控制台会话的权限

AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 的权限

所担任的角色的权限策略决定由 AssumeRole、AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity 返回的临时安全凭证的权限。您可在创建或更新该角色时定义这些权限。

（可选）您可以将内联或托管会话策略作为 AssumeRole、AssumeRoleWithSAML 或 AssumeRoleWithWebIdentity API 操作的参数传递。会话策略限制角色的临时凭证会话的权限。生成的会话的权限是角色的基于身份的策略与会话策略的交集。您可以在后续的 Amazon API 调用中使用角色的临时凭证来访问拥有该角色的账户中的资源。您使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。要了解有关 Amazon 如何确定角色的有效权限的更多信息，请参阅策略评估逻辑。



在做出“允许”或“拒绝”授权决定时，Amazon 不会对附加至发出 AssumeRole 原始调用的凭证的策略进行评估。用户将临时放弃其原始权限，以支持担任的角色所分配的权限。对于 AssumeRoleWithSAML 和 AssumeRoleWithWebIdentity API 操作，不会有任何策略受到评估，因为这些 API 的发起人不是 Amazon 身份。

示例：使用 AssumeRole 分配权限

您可以将 AssumeRole API 操作与不同类型的策略结合使用。以下是一些示例。

角色权限策略

在本示例中，您调用 AssumeRole API 操作，而无需在可选 Policy 参数中指定会话策略。分配给临时凭证的权限取决于所担任角色的权限策略。以下示例权限策略向角色授予权限，允许该角色列出名为 productionapp 的 S3 存储桶中包含的所有对象。它还允许该角色获取、放置和删除该存储桶中的对象。

Example角色权限策略示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

作为参数传递的会话策略

假设您希望允许某用户担任上例中的相同角色。但在这种情况下，您希望角色会话只具有在 productionapp S3 存储桶中获取和放入对象的权限。您不希望允许他们删除对象。完成该操作的一种方法是创建一个新角色，并在该角色的权限策略中指定所需的权限。完成该操作的另一种方法是，调用 AssumeRole API 并在可选的 Policy 参数中包含会话策略以作为 API 操作的一部分。生成的会话的权限是角色的基于身份的策略与会话策略的交集。使用会话策略授予的权限不能超过担任的角色的基于身份的策略允许的权限。有关角色会话权限的更多信息，请参阅会话策略。

在检索新会话的临时凭证后，您可以将其传递给希望具有这些权限的用户。

例如，假设将以下策略作为该 API 调用的参数传递。使用会话的用户只具备执行以下操作的权限：

- 列出 productionapp 存储桶中的所有对象。
- 获取 productionapp 存储桶中的对象或向其中上传对象。

在以下会话策略中，s3:DeleteObject 权限已被筛选掉，因此，未向担任的会话授予 s3:DeleteObject 权限。该策略为角色会话设置最大权限，以便它覆盖角色上的任何现有权限策略。

Example通过 AssumeRole API 调用传递的示例会话策略

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::productionapp"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::productionapp/*"
    }
  ]
}
```

基于资源的策略

某些 Amazon 资源支持基于资源的策略，这些策略提供另一种机制用于定义影响临时安全凭证的权限。仅几种资源（如 Amazon S3 存储桶、Amazon SNS 主题和 Amazon SQS 队列）支持基于资源的策略。下面的示例使用名为 productionapp 的 S3 存储桶进一步阐述上述示例。以下策略被附加到该存储桶。

在将以下基于资源的策略附加到 productionapp 存储桶时，将会拒绝所有 用户从该存储桶中删除对象的权限。（请参阅策略中的 Principal 元素。）这包括所有担任角色的用户（即使角色权限策略授予了 DeleteObject 权限）。显式 Deny 语句总是优先于 Allow 语句。

Example存储桶策略的示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Principal": {"AWS": "*"},
    "Effect": "Deny",
    "Action": "s3:DeleteObject",
    "Resource": "arn:aws:s3:::productionapp/*"
  }
}
```

有关 Amazon 如何对多个策略类型进行合并和评估的更多信息，请参阅策略评估逻辑。

监控和控制使用所担任角色执行的操作

IAM 角色是 IAM 中分配权限的对象。当您使用 IAM 身份或来自 Amazon 以外的身份担任该角色时，将会收到具有分配给该角色的权限的会话。

当您在 Amazon 中执行操作时，则有关您会话的信息可以记录到 Amazon CloudTrail 以便您的账户管理员进行监控。管理员可以配置角色以要求身份来传递自定义字符串，该字符串标识在 Amazon 中执行操作的个人或应用程序，称为源身份。此身份信息在 Amazon CloudTrail 中存储为源身份。当管理员查看 CloudTrail 中的活动时，他们可以查看源身份信息，以确定谁通过担任角色会话执行了操作或执行了哪些操作。

设置源身份后，它会出现在任何在角色会话期间执行的 Amazon 操作的请求中。当角色用于通过 Amazon CLI 或者 Amazon API 担任其他角色时，设置的值仍然存在，这称为角色链。在角色会话期间无法更改已设置的值。管理员可以根据源身份的存在或值配置精细权限，以进一步控制使用共享角色执行的 Amazon 操作。您可以决定是否可以使用源身份属性、是否需要该属性以及可以使用哪些值。

使用源身份的方式与使用角色会话名称和会话标签的方式大有不同。设置后，源身份值无法更改，并且对角色会话执行的任何其他操作将持续存在。以下是如何使用会话标签和角色会话名称的说明：

- Session tags（会话标签）- 您也可以在代入角色或联合身份用户时传递会话标签。担任角色时会出现会话标签。您可以定义策略，这些策略使用标签条件键来根据主体的标签向其授予权限。然后您可以使用 CloudTrail 查看为代入角色或联合身份用户而发出的请求。要了解会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。
- Role session name（角色会话名称）- 您可以在角色信任策略中使用 `sts:RoleSessionName` 条件键，以要求您的用户在代入角色时提供特定的会话名称。角色会话名称可用于区分角色由不同主体使用时的角色会话。要了解角色会话名称的更多信息，请参阅 sts:RoleSessionName。

我们建议您在要控制担任角色的身份时使用源身份。源身份在挖掘 CloudTrail 日志以确定是何人在使用该角色执行操作时也很有用。

主题

- 设置以使用源身份
- 有关源身份的需知信息
- 设置源身份时所需的权限
- 在担任角色时指定源身份
- 使用具有 AssumeRole 的源身份
- 使用具有 AssumeRoleWithSAML 的源身份
- 使用具有 AssumeRoleWithWebIdentity 源身份
- 使用源身份信息控制访问
- 在 CloudTrail 中查看源身份

设置以使用源身份

设置以使用源身份的方式取决于担任角色时使用的方法。例如，您的 IAM 用户可以直接使用 `AssumeRole` 操作。如果您具有企业身份（也称为人力身份），则他们可能会使用 `AssumeRoleWithSAML` 访问您的 Amazon 资源。如果终端用户访问您的移动应用程序或 Web 应用程序，他们可能会使用 `AssumeRoleWithWebIdentity`。以下是一个高级别的工作流概览，可帮助您了解如何进行设置以在现有环境中利用源身份信息。

1. 配置测试用户和角色 — 使用预生产环境，配置测试用户和角色，并配置其策略以允许设置源身份。

   如果您对联合身份使用身份提供程序 (IdP)，请将 IdP 配置为在断言或令牌中传递您选择的源身份的用户属性。
2. 担任角色 — 测试所担任角色并将源身份传递给您为测试而设置的用户和角色。

3. 查看 CloudTrail — 查看 CloudTrail 日志中测试角色的源身份信息。
4. 培训您的用户 — 在预生产环境中进行测试后，请确保用户知道如何传递源身份信息（如有必要）。设置要求用户在生产环境中提供源身份的最后期限。
5. 配置生产策略 — 为生产环境配置策略，然后将其添加到生产用户和角色中。
6. 监控活动 — 使用 CloudTrail 日志监控您的生产角色活动。

有关源身份的需知信息

使用源身份时请记住以下事项。

- 连接到身份提供程序 (IdP) 的所有角色的角色信任策略都必须具有 `sts:SetSourceIdentity` 权限。对于信任策略中没有此权限的角色，`AssumeRole*` 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递源身份。然后仅将 `sts:SetSourceIdentity` 权限添加到连接至单独 IdP 的角色。
- 当身份设置源身份时，`sts:SourceIdentity` 密钥将出现在请求中。对于角色会话期间执行的后续操作，`aws:SourceIdentity` 密钥将出现在请求中。Amazon 不控制 `sts:SourceIdentity` 或 `aws:SourceIdentity` 密钥中源身份的值。如果选择要求源身份，则必须选择希望用户或 IdP 提供的属性。出于安全考虑，您必须确保可以控制如何来提供这些值。
- 源身份值的长度必须介于 2 到 64 个字符之间，只能包含字母数字字符、下划线和以下字符：. , + = @ - （连字符）。您不能创建以文本 **`aws:`** 开头的值。此前缀是专为 Amazon 内部使用预留的。
- 当 Amazon 服务或服务关联角色代表联合身份或工作人员身份执行操作时，源身份信息不会被 CloudTrail 捕获。

> **Important**
>
> 您无法切换为 Amazon Web Services 管理控制台 中在担任角色时需要设置源身份的角色。要担任这样的角色，您可以使用 Amazon CLI 或者 Amazon API 来调用 `AssumeRole` 操作并指定源身份参数。

设置源身份时所需的权限

除了与 API 操作匹配的操作之外，您的策略中还必须具有以下仅限授权执行的操作：

```
sts:SetSourceIdentity
```

- 要指定源身份，主体（IAM 用户和角色）必须具有对 sts:SetSourceIdentity 的权限。作为管理员，您可以在角色信任策略和主体的权限策略中对此进行配置。
- 当您使用另一个角色担任角色时，该操作称为角色链，在担任角色的主体的权限策略和目标角色的角色信任策略中都需要对 sts:SetSourceIdentity 的权限。否则，担任角色的操作将失败。
- 使用源身份时，连接到 IdP 的所有角色的角色信任策略都必须具有 sts:SetSourceIdentity 权限。对于连接到 IdP 的任何角色，如果在没有此权限的情况下，AssumeRole* 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递源身份，然后将 sts:SetSourceIdentity 权限仅添加到与单独的 IdP 连接的角色。
- 要跨账户边界设置源身份，您必须在两个位置包含 sts:SetSourceIdentity 权限。该权限必须位于原始账户中主体的权限策略和目标账户中角色的角色信任策略中。例如，当某个角色用于在另一个账户中使用角色链担任某个角色时，您可能需要执行此操作。

作为账户管理员，假设您希望允许 IAM 用户 DevUser 在您的账户中担任在同一账户中的 Developer_Role。但是，您希望只有当用户已将源身份设置为其 IAM 用户名时才允许此操作。那么，您可以将以下策略附加到 IAM 用户。

Example附加到 DevUser 的基于身份的策略示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRole",
      "Effect": "Allow",
      "Action": "sts:AssumeRole",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role"
    },
    {
      "Sid": "SetAwsUserNameAsSourceIdentity",
      "Effect": "Allow",
      "Action": "sts:SetSourceIdentity",
      "Resource": "arn:aws:iam::123456789012:role/Developer_Role",
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": "${aws:username}"
        }
      }
```

```
    }
  ]
}
```

要强制实施可接受的源身份值，可以配置以下角色信任策略。该策略为 IAM 用户提供 DevUser 权限以担任该角色并设置源身份。sts:SourceIdentity 条件键定义可接受的源身份值。

Example源身份的角色信任策略示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowDevUserAssumeRole",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:user/DevUser"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringEquals": {
          "sts:SourceIdentity": "DevUser"
        }
      }
    }
  ]
}
```

借助 IAM 用户 DevUser 的凭证，用户将尝试担任使用以下 Amazon CLI 请求的 DeveloperRole。

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Developer_Role \
--role-session-name Dev-project \
```

```
--source-identity DevUser \
```

当 Amazon 评估请求时，请求上下文将包含 DevUser 的 sts:SourceIdentity。

在担任角色时指定源身份

当您使用 Amazon STS AssumeRole*API 操作来获取角色的临时安全凭证时，可以指定源身份。您使用的 API 操作因您的使用案例而异。例如，如果您使用 IAM 角色授予 IAM 用户访问 Amazon 资源的权限（而其通常并不具有此权限），则可以使用 AssumeRole 操作。如果您使用企业联合身份验证来管理工作人员用户，则可以使用 AssumeRoleWithSAML 操作。如果您使用 OIDC 联合身份验证来允许终端用户访问您的移动或 Web 应用程序，则可以使用 AssumeRoleWithWebIdentity 操作。以下部分介绍如何在每个操作中使用源身份。要了解有关临时凭证常见场景的更多信息，请参阅 临时凭证的常见情形。

使用具有 AssumeRole 的源身份

AssumeRole 操作返回一组可用于访问 Amazon 资源的临时凭证。您可以使用 IAM 用户或角色凭证来调用 AssumeRole。要在代入角色时传递源身份，请使用 --source-identity Amazon CLI 选项或 SourceIdentity Amazon API 参数。以下示例说明如何使用 Amazon CLI 指定源身份。

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/developer \
--role-session-name Audit \
--source-identity Admin \
```

使用具有 AssumeRoleWithSAML 的源身份

主体调用 AssumeRoleWithSAML 操作使用基于 SAML 的联合身份进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 SAML 的联合身份用于 Amazon Web Services 管理控制台访问的更多信息，请参阅使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。有关 Amazon CLI 或 Amazon API 访问的详细信息，请参阅SAML 2.0 联合身份验证。有关为 Active Directory 用户设置 SAML 联合身份的教程，请参阅 Amazon 安全博客中的使用 Active Directory Amazon 联合身份验证服务 (ADFS) 的亚马逊云科技联合身份验证。

作为管理员，您可以使用 Amazon STS AssumeRoleWithSAML 操作，允许公司目录的成员联合身份到 Amazon 中。要执行此操作，您必须完成以下任务：

1. 在企业中配置 SAML 提供程序。
2. 在 IAM 中创建 SAML 提供商
3. 在 Amazon 中为 SAML 联合主体配置角色及其权限。
4. 完成配置 SAML IdP 并为 SAML 身份验证响应创建断言。

要为源身份设置 SAML 属性，请包含 Attribute 元素，并将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/SourceIdentity。使用 AttributeValue 元素指定源身份的值。例如，假设您要将以下身份属性作为源身份传递。

```
SourceIdentity:DiegoRamirez
```

要传递这些属性，请在 SAML 断言中包含以下元素。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/SourceIdentity">
<AttributeValue>DiegoRamirez</AttributeValue>
</Attribute>
```

使用具有 AssumeRoleWithWebIdentity 源身份

调用 AssumeRoleWithWebIdentity 操作的主体使用兼容 OpenID Connect（OIDC）的联合身份验证进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将 OIDC 联合身份验证用于 Amazon Web Services 管理控制台访问的更多信息，请参阅 OIDC 联合身份验证。

要从 OpenID Connect (OIDC) 传递源身份，您必须在 JSON Web Token (JWT) 中包含源身份。当您提交 AssumeRoleWithWebIdentity 请求时，请在令牌的 https://www.amazonaws.cn/ source_identity 命名空间中包含源身份。要了解有关 OIDC 令牌和声明的更多信息，请参阅《Amazon Cognito 开发人员指南》中的将令牌与用户群体结合使用。

例如，以下解码的 JWT 是一个令牌，用于通过 Admin 源身份调用 AssumeRoleWithWebIdentity。

Example解码的 JSON Web 令牌示例

```
{
    "sub": "john",
    "aud": "ac_oic_client",
```

```
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/source_identity":"Admin"
}
```

使用源身份信息控制访问

当初始设置源身份时，sts:SourceIdentity 密钥将出现在请求中。设置源身份后，aws:SourceIdentity 密钥存在于角色会话期间提出的所有后续请求中。作为管理员，您可以编写策略，授予条件授权以根据源身份属性的现状或值执行 Amazon 操作。

假设您希望要求开发人员设置源身份，以承担一个关键角色，该角色有权写入生产关键 Amazon 资源。再假设您使用 AssumeRoleWithSAML 向您的工作人员身份授予了 Amazon 访问权限。您只希望高级开发人员 Saanvi 和 Diego 具有该角色的访问权限，因此您可以为角色创建以下信任策略。

Example源身份的角色信任策略示例 (SAML)

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "SAMLProviderAssumeRoleWithSAML",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
        "sts:AssumeRoleWithSAML"
      ],
      "Condition": {
        "StringEquals": {
          "SAML:aud": "https://signin.aws.amazon.com/saml"
        }
      }
    },
    {
```

```
      "Sid": "SetSourceIdentitySrEngs",
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:saml-provider/name-of-identity-
provider"
      },
      "Action": [
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

信任策略包含 sts:SourceIdentity 的条件，需要将源身份设置为能够使 Saanvi 或 Diego 担任关键角色。

或者，如果您使用 OIDC 提供者进行联合身份验证，并用 AssumeRoleWithWebIdentity 对用户进行身份验证，您的角色信任策略可能如下所示。

Example源身份的角色信任策略示例（OIDC 提供商）

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/server.example.com"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:SetSourceIdentity"
```

```
      ],
      "Condition": {
        "StringEquals": {
          "server.example.com:aud": "oidc-audience-id"
        },
        "StringLike": {
          "sts:SourceIdentity": [
            "Saanvi",
            "Diego"
          ]
        }
      }
    }
  ]
}
```

角色链和跨账户要求

假设您想允许已经担任 CriticalRole 的用户在另一个账户中担任 CriticalRole_2。获得以用户担任 CriticalRole 的角色会话凭证用于角色链到不同账户中的第二个角色，CriticalRole_2。该角色正在以跨账户边界的形式担任。因此，sts:SetSourceIdentity 权限必须在权限策略中授予 CriticalRole，在角色信任策略中授予 CriticalRole_2。

Example CriticalRole 上的权限策略示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AssumeRoleAndSetSourceIdentity",
      "Effect": "Allow",
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Resource": "arn:aws:iam::222222222222:role/CriticalRole_2"
    }
  ]
}
```

为确保跨账户边界设置源身份，以下角色信任策略仅信任 CriticalRole 的角色主体来设置源身份。

Example CriticalRole_2 上的角色信任策略示例

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/CriticalRole"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:SetSourceIdentity"
      ],
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": ["Saanvi","Diego"]
        }
      }
    }
  ]
}
```

用户使用从担任 CriticalRole 获得的角色会话凭证进行以下调用。源身份是在担任 CriticalRole 期间设置的，因此无需再次显式设置。如果用户尝试设置与担任 CriticalRole 所设置值不同的源身份，则担任角色请求将被拒绝。

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::222222222222:role/CriticalRole_2 \
--role-session-name Audit \
```

当调用主体担任角色时，请求中的源身份将从第一个担任的角色会话保留。因此，aws:SourceIdentity 和 sts:SourceIdentity 密钥将会出现在请求上下文中。

在 CloudTrail 中查看源身份

您可以使用 CloudTrail 查看为代入角色或联合身份用户而发出的请求。您还可以查看角色或用户请求，以便在 Amazon 中执行操作。CloudTrail 日志文件包括有关所代入角色或联合身份用户会话的源身份集的信息。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。

例如，假设用户提出 Amazon STS `AssumeRole` 请求，并设置了源身份。您可以在 CloudTrail 日志的 `requestParameters` 键中找到 `sourceIdentity` 信息。

Example Amazon CloudTrail 日志中的示例 requestParameters 部分

```
"eventVersion": "1.05",
    "userIdentity": {
        "type": "AWSAccount",
        "principalId": "AIDAJ45Q7YFFAREXAMPLE",
        "accountId": "111122223333"
    },
    "eventTime": "2020-04-02T18:20:53Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "203.0.113.64",
    "userAgent": "aws-cli/1.16.96 Python/3.6.0 Windows/10 botocore/1.12.86",
    "requestParameters": {
        "roleArn": "arn:aws:iam::123456789012:role/DevRole",
        "roleSessionName": "Dev1",
        "sourceIdentity": "source-identity-value-set"
    }
```

如果用户使用所担任的角色会话执行操作，则源身份信息将出现在 CloudTrail 日志的 `userIdentity` 密钥中。

Example Amazon CloudTrail 日志中的 userIdentity 密钥示例

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAJ45Q7YFFAREXAMPLE:Dev1",
    "arn": "arn:aws:sts::123456789012:assumed-role/DevRole/Dev1",
    "accountId": "123456789012",
    "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
```

```
    "sessionContext": {
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAJ45Q7YFFAREXAMPLE",
        "arn": "arn:aws:iam::123456789012:role/DevRole",
        "accountId": "123456789012",
        "userName": "DevRole"
      },
      "webIdFederationData": {},
      "attributes": {
        "mfaAuthenticated": "false",
        "creationDate": "2021-02-21T23:46:28Z"
      },
      "sourceIdentity": "source-identity-value-present"
    }
  }
}
```

要查看 CloudTrail 日志中的示例 Amazon STS API 事件，请参阅 CloudTrail 日志中的 IAM API 事件示例。有关 CloudTrail 日志文件中所包含信息的更多详细信息，请参阅 Amazon CloudTrail 用户指南中的 CloudTrail 事件参考。

GetFederationToken 的权限

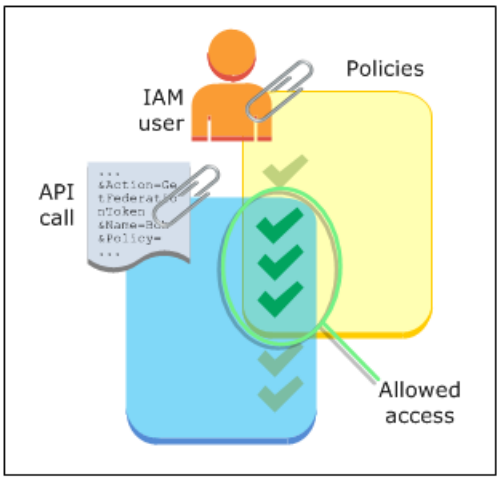
GetFederationToken 操作是由 IAM 用户调用的，并返回该用户的临时凭证。该操作联合 用户。分配给 Amazon STS 联合用户会话的权限是在以下两个位置之一中定义的：

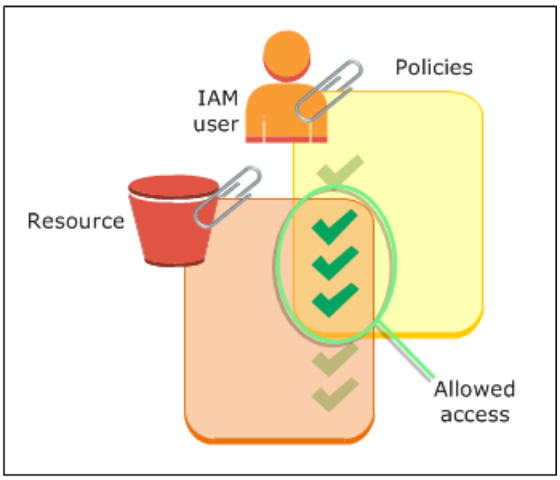
- 作为 GetFederationToken API 调用参数传递的会话策略。(最常见。)
- 一项基于资源的策略，在该策略的 Principal 元素中对 Amazon STS 联合用户会话进行显式命名。（不太常见。）

会话策略是高级策略，在以编程方式创建临时会话时，这些策略将作为参数进行传递。在创建 Amazon STS 联合用户会话并传递会话策略时，生成的会话的权限是用户的基于身份的策略与会话策略的交集。您使用会话策略授予的权限不能超过联合的用户的基于身份的策略允许的权限。

在大多数情况下，如果您未通过 GetFederationToken API 调用传递策略，则生成的临时安全凭证没有任何权限。不过，基于资源的策略可以为会话提供额外的权限。您可以使用将会话指定为允许的主体的基于资源的策略访问资源。

下图形象地展示了各策略之间如何相互影响，以确定调用 GetFederationToken 所返回的临时安全凭证的权限。





示例：使用 GetFederationToken 分配权限

您可以将 GetFederationToken API 操作与不同类型的策略结合使用。以下是一些示例。

策略已附加到 IAM 用户

在此示例中，您拥有一个基于浏览器的客户端应用程序，该程序依赖于两个后端 Web 服务。一个后端服务是您自己的身份验证服务器，该服务器使用您自己的身份系统对客户端应用程序进行验证。另一个后端服务是一项 Amazon 服务，该服务用于提供此客户端应用程序的部分功能。您的服务器对该客户端应用程序进行身份验证，并创建或检索相应的权限策略。之后，您的服务器调用 GetFederationToken API 来获取临时安全凭证，并将这些凭证返回给客户端应用程序。接下来，客户端应用程序就可以借助该临时安全凭证向 Amazon 服务直接发出请求。这一架构允许客户端应用程序在不嵌入长期 Amazon 凭证的情况下发出 Amazon 请求。

您的身份验证服务器使用名为 token-app 的 IAM 用户的长期安全凭证调用 GetFederationToken API。但是，长期 IAM 用户凭证保留在您的服务器上，从不分发到客户端。下面的示例策略将被附加至 token-app IAM 用户，其中定义了您的 Amazon STS 联合用户（客户端）需要的一组最广泛的权限。请注意，您的身份验证服务器需要 sts:GetFederationToken 权限才能获取 Amazon STS 联合用户的临时安全凭证。

Example附加到 IAM 用户 token-app 并调用 GetFederationToken 的示例策略

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "sts:GetFederationToken",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "dynamodb:ListTables",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sqs:ReceiveMessage",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "sns:ListSubscriptions",
      "Resource": "*"
    }
  ]
}
```

上述策略为 IAM 用户授予多个权限。不过，该策略本身不会为 Amazon STS 联合用户授予任何权限。如果该 IAM 用户调用 `GetFederationToken`，并且未将策略作为该 API 调用的参数传递，则生成的 Amazon STS 联合用户没有任何有效的权限。

作为参数传递的会话策略

要确保为 Amazon STS 联合用户分配相应的权限，最常用的方法是在 `GetFederationToken` API 调用中传递会话策略。让我们进一步阐述上述示例，并假设使用 IAM 用户 `token-app` 的凭证调用了 `GetFederationToken`。然后，假设将以下会话策略作为该 API 调用的参数进行传递。生成的 Amazon STS 联合用户有权列出名为 `productionapp` 的 Amazon S3 存储桶的内容。用户无法对 `productionapp` 存储桶中的项目执行 Amazon S3 `GetObject`、`PutObject`、和 `DeleteObject` 操作。

将为联合身份用户分配这些权限，因为这些权限是 IAM 用户策略与您传递的会话策略的交集。

Amazon STS 联合用户无法在 Amazon SNS、Amazon SQS、Amazon DynamoDB，或任何 S3 存储桶（`productionapp` 除外）中执行操作。这些操作会被拒绝，即使这些权限被授给了与 `GetFederationToken` 调用关联的 IAM 用户。

Example作为 GetFederationToken API 调用参数传递的示例会话策略

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::productionapp"]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": ["arn:aws:s3:::productionapp/*"]
    }
  ]
}
```

基于资源的策略

某些 Amazon 资源支持基于资源的策略，这些策略提供了另一种直接向 Amazon STS 联合用户授权的机制。只有一部分 Amazon 服务支持基于资源的策略。例如，Amazon S3 具有存储桶，Amazon SNS 有主题，Amazon SQS 具有队列，您可以向这些内容附加策略。有关支持基于资源的策略的所有服务的列表，请参阅使用 IAM 的 Amazon 服务并回顾表的“基于资源的策略”列。您可以使用基于资源的策略将权限直接分配给 Amazon STS 联合用户。为此，请在基于资源的策略的 `Principal` 元素中指定 Amazon STS 联合用户的 Amazon 资源名称（ARN）。以下示例说明了这一点，并使用名为 `productionapp` 的 S3 存储桶进一步阐述上述示例。

以下基于资源的策略附加到存储桶。此存储桶策略允许名为 Carol 的 Amazon STS 联合用户访问该存储桶。当之前介绍的示例策略被附加至 `token-app` IAM 用户时，名为 Carol 的 Amazon STS

联合用户将具有对名为 productionapp 的存储桶执行 s3:GetObject、s3:PutObject 和 s3:DeleteObject 操作的权限。即使未将任何会话策略作为 GetFederationToken API 调用的参数传递，Carol 也能够执行此操作。这是因为，在此示例中，以下基于资源的策略已向名为 Carol 的 Amazon STS 联合用户明确授予权限。

请记住，只有向 IAM 用户和 Amazon STS 联合用户都进行显式授权时，Amazon STS 联合用户才具有这些权限。另外，也可通过在策略的 Principal 元素中对 Amazon STS 联合用户进行显式命名的基于资源的策略授予这些权限（在账户内），如下例所示。

Example允许访问联合身份用户的示例存储桶策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Principal": {
            "AWS": "arn:aws:sts::111122223333:federated-user/Carol"
        },
        "Effect": "Allow",
        "Action": [
            "s3:GetObject",
            "s3:PutObject",
            "s3:DeleteObject"
        ],
        "Resource": [
            "arn:aws:s3:::productionapp/*"
        ]
    }
}
```

有关如何评估策略的更多信息，请参阅策略评估逻辑。

GetSessionToken 的权限

调用 GetSessionToken API 操作或 get-session-token CLI 命令主要发生在用户必须使用 Multi-Factor Authentication (MFA) 进行身份验证时。可以编写一条策略，只允许特定操作，且仅当这些操作是由经过 MFA 身份验证的用户请求时，才予以放行。为成功通过 MFA 身份验证检查，用户必须先调用 GetSessionToken，并包含可选的 SerialNumber 和 TokenCode 参数。如果用户成功通过

MFA 设备的身份验证，则 GetSessionToken API 调用返回的凭证将包含 MFA 上下文。此上下文指示用户已使用 MFA 进行了身份验证，并已获得了需要 MFA 身份验证的 API 操作的授权。

GetSessionToken 所需的权限

用户无需任何权限即可获取会话令牌。GetSessionToken 操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证操作。

要授予执行大多数 Amazon 操作的权限，您可以将具有相同名称的操作添加到策略。例如，要创建用户，您必须使用 CreateUser API 操作、create-user CLI 命令或 Amazon Web Services 管理控制台。要执行这些操作，您必须具有一个策略，该策略允许您访问 CreateUser 操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateUser",
            "Resource": "*"
        }
    ]
}
```

虽然您可以在策略中包含 GetSessionToken 操作，但不会影响用户执行 GetSessionToken 操作的能力。

由 GetSessionToken 授予的权限

如果调用 GetSessionToken 时使用的是 IAM 用户的凭证，则临时安全凭证将具有与该 IAM 用户相同的权限。同样，如果使用 Amazon Web Services 账户根用户 凭证调用 GetSessionToken，临时安全凭证将拥有根用户权限。

> **Note**
>
> 我们建议您不要使用根用户凭证来调用 GetSessionToken。相反，请遵循我们的最佳实践，并创建具有所需权限的 IAM 用户。然后使用这些 IAM 用户执行与 Amazon 的日常交互工作。

您在调用 GetSessionToken 时获得的临时凭证具有以下功能和限制：

- 您可以通过将凭证传递到 https://signin.aws.amazon.com/federation 上的联合身份验证单一登录终端节点来访问 Amazon Web Services 管理控制台。有关更多信息，请参阅 使自定义身份凭证代理程序能够访问 Amazon 控制台。
- 您无法使用凭证调用 IAM 或 Amazon STS API 操作。您可以使用它们来调用其他 Amazon 服务的 API 操作。

请参阅 比较 Amazon STS 凭证，将此 API 操作及其限制和功能与创建临时安全凭证的其他 API 操作比较

有关使用 GetSessionToken 进行受 MFA 保护的 API 访问的更多信息，请参阅使用 MFA 保护 API 访问。

禁用临时安全凭证的权限

临时安全凭证在过期之前一直有效。这些凭证在指定的时间段内有效，即 900 秒（15 分钟）到最长 129600 秒（36 小时）之间。默认会话持续时间为 43200 秒（12 小时）。您可以撤消这些凭证，但还须更改 IAM 用户或角色的权限，以阻止他人通过泄露的凭证进行恶意账户活动。每次使用临时安全凭证发出 Amazon 请求时，系统都会评估分配给该凭证的权限。从凭证中删除所有权限后，使用这些权限的 Amazon 请求会失败。

可能需要几分钟时间，策略更新才能生效。对于 IAM 角色会话，可以撤消角色的临时安全凭证，以强制担任该角色的所有用户重新进行身份验证并请求新的凭证。有关更多信息，请参阅撤销 IAM 角色临时安全凭证。

您无法更改 Amazon Web Services 账户根用户的权限。同样，以根用户身份登录时，您无法更改调用 GetFederationToken 或 GetSessionToken 创建的临时安全凭证的权限。因此，我们建议您不要以根用户身份调用 GetFederationToken 或 GetSessionToken。

有关如何更改 IAM 用户的权限的过程，请参阅更改 IAM 用户的权限。

有关如何更改 IAM 角色的权限的过程，请参阅更新角色的权限。

> **Important**
>
> 您不能编辑 IAM 中根据 IAM Identity Center 权限集创建的角色。您必须在 IAM Identity Center 中撤消用户的活动权限集会话。有关更多信息，请参阅《IAM Identity Center 用户指南》中的撤消由权限集创建的活动 IAM 角色会话。

主题

- 拒绝访问与角色关联的所有 IAM 角色会话
- 拒绝访问特定 IAM 角色会话
- 使用条件上下文键拒绝访问临时安全凭证会话
- 拒绝访问使用基于资源的策略的指定主体

拒绝访问与角色关联的所有 IAM 角色会话

此过程拒绝针对与角色关联的所有 IAM 角色会话的权限。如果担心他人通过以下方式进行可疑访问，可使用此方法：

- 使用跨账户存取权限的其他账户主体
- 有权访问账户中 Amazon 资源的外部用户身份
- 已在移动或 Web 应用程序中使用 OIDC 提供者进行身份验证的用户

要更改或删除通过调用 `AssumeRole`、`AssumeRoleWithSAML`、`AssumeRoleWithWebIdentity`、`GetFederationToken` 或 `GetSessionToken` 而获取的临时安全凭证分配的权限，可编辑或删除为角色定义权限的基于身份的策略。

> **Important**
>
> 如果存在允许主体访问的基于资源的策略，您还必须为该资源添加显式拒绝。有关详细信息，请参阅 拒绝访问使用基于资源的策略的指定主体。

拒绝访问与角色关联的所有 IAM 角色会话

1. 登录到 Amazon Web Services 管理控制台 并打开 IAM 控制台。
2. 在导航窗格中，选择角色。
3. 选择要编辑的角色的名称。您可以使用搜索框来筛选列表。
4. 选择权限选项卡。
5. 选择要编辑的相关策略。在编辑客户管理型策略之前，请查看附加的实体选项卡，以免中断用户对可能附加了相同策略的其他身份的访问权限。

6. 选择 JSON 选项卡并更新策略以拒绝所有资源和操作。

> **Note**
>
> 这些权限与 Amazon 托管式策略 AWSDenyAll 中的权限相同。您可以将此 Amazon 托管式策略附加到您想要拒绝其所有访问权限的任何 IAM 用户或角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAll",
            "Effect": "Deny",
            "Action": [
                "*"
            ],
            "Resource": "*"
        }
    ]
}
```

7. 在审核页面上，检查策略摘要，然后选择保存更改进行保存。

更新策略时，所做的更改会影响与该角色关联的所有临时安全凭证的权限，包括在更改该角色的权限策略之前颁发的凭证。

更新策略后，您可以撤消角色的临时安全凭证，以立即撤消对该角色的已颁发凭证的所有权限。

拒绝访问特定 IAM 角色会话

当您使用“拒绝所有”策略更新 IAM 角色或完全删除该角色时，所有有权访问该角色的用户的访问权限都会中断。您可以在不影响与角色关联的所有其他会话的权限的情况下拒绝访问。

可以使用条件上下文键或基于资源的策略拒绝执行 `Principal` 的权限。

> **Tip**
>
> 您可以使用 Amazon CloudTrail 日志查找联合用户的 ARN。有关更多信息，请参阅 How to Easily Identify Your Federated Users by Using Amazon CloudTrail。

使用条件上下文键拒绝访问临时安全凭证会话

如果您想要拒绝访问特定临时安全凭证会话，而不影响创建该凭证的 IAM 用户或角色的权限，可以在基于身份的策略中使用条件上下文键。对于 IAM 角色，更新策略后，您可以撤消角色的临时安全凭证会话，以立即撤消所有已颁发的凭证。

有关条件上下文键的更多信息，请参阅 Amazon 全局条件上下文密钥。

aws:PrincipalArn

您可以在基于身份的策略中使用条件上下文键 aws:PrincipalArn，根据 Amazon 资源名称（ARN）来拒绝访问特定主体。为此，您可以在策略的 Condition 元素中指定与临时安全凭证关联的 IAM 用户、角色或 Amazon STS 联合用户会话。

根据 ARN 拒绝访问特定主体

1. 在 IAM 控制台的导航窗格中，选择用户或角色。
2. 选择要编辑的 IAM 用户或角色的名称。您可以使用搜索框来筛选列表。
3. 选择权限选项卡。
4. 选择要编辑的相关策略。在编辑客户管理型策略之前，请查看附加的实体选项卡，以免中断用户对可能附加了相同策略的其他身份的访问权限。
5. 选择 JSON 选项卡并为主体 ARN 添加拒绝语句，如以下示例所示：

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
```

```
      "Condition": {
        "ArnEquals": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:role/ROLENAME",
            "arn:aws:iam::222222222222:user/USERNAME",
            "arn:aws:iam::222222222222:federated-user/USERNAME"
          ]
        }
      }
    }
  ]
}
```

6. 在审核页面上，检查策略摘要，然后选择保存更改进行保存。

aws:SourceIdentity

您可以在基于身份的策略中使用条件上下文键 aws:SourceIdentity 来拒绝访问与 IAM 角色会话关联的特定源身份。只要在主体使用任何 Amazon STS `assume-role*` CLI 命令或 Amazon STS `AssumeRole*` API 操作担任角色时通过设置 `SourceIdentity` 请求参数发出角色会话，此方法就适用。您可以通过在策略的 `Condition` 元素中指定与临时安全凭证关联的源身份来执行此操作。

与上下文键 sts:RoleSessionName 不同，在设置源身份后，无法更改该值。`aws:SourceIdentity` 键存在于角色执行的所有操作的请求上下文中。当您使用会话凭证担任另一个角色时，源身份将保留到后续角色会话中。从一个角色代入另一个角色的过程称为角色链。

以下策略显示了如何使用条件上下文键 `aws:SourceIdentity` 拒绝访问临时安全凭证会话的示例。如果您指定与角色会话关联的源身份，则将拒绝具有已命名源身份的角色会话，而不会影响已创建凭证的角色的权限。对于此示例，发出角色会话时主体设置的源身份为 `nikki_wolf@example.com`。源身份为 `nikki_wolf@example.com` 的角色会话发出的任何请求都将被拒绝，因为源身份包含在策略条件中，并且策略效果设置为 `Deny`。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
```

```
      "Condition": {
        "StringLike": {
          "aws:SourceIdentity": [
            "nikki_wolf@example.com",
            "<source identity value>"
          ]
        }
      }
    }
  ]
}
```

aws:userid

您可以在基于身份的策略中使用条件上下文键 aws:userid 来拒绝访问与 IAM 用户或角色关联的所有或特定临时安全凭证会话。为此，您可以在策略的 Condition 元素中指定与临时安全凭证关联的 IAM 用户、角色或 Amazon STS 联合用户会话的唯一标识符（ID）。

以下策略显示了如何使用条件上下文键 aws:userid 拒绝访问临时安全凭证会话的示例。

- AIDAXUSER1 表示 IAM 用户的唯一 ID。如果将 IAM 用户的唯一 ID 指定为上下文键 aws:userid 的值，这将拒绝访问该 IAM 用户。这包括通过调用 GetSessionToken API 创建的任何临时安全凭证会话。
- AROAXROLE1:* 表示与 IAM 角色关联的所有会话的唯一 ID。如果在 caller-specified-role-session-name 部分中将 IAM 角色的唯一 ID 和通配符 (*) 指定为上下文键 aws:userid 的值，这将拒绝与该角色关联的所有会话。
- AROAXROLE2:<caller-specified-role-session-name> 表示担任角色的会话的唯一 ID。在担任角色的唯一 ID 的 caller-specified-role-session-name 部分中，如果使用 StringLike 条件运算符，则可指定角色会话名称或通配符。如果您指定角色会话名称，则将拒绝已命名的角色会话，而不会影响已创建凭证的角色的权限。如果您为角色会话名称指定通配符，则将拒绝与该角色关联的所有会话。

> **Note**
>
> 调用者指定的角色会话名称，是担任角色会话唯一标识符的一部分，在角色链接期间可能会发生变化。当一个角色担任另一个角色时，就会发生角色链接。角色会话名称是在主体使用 Amazon STS AssumeRole API 操作担任角色时使用 RoleSessionName 请求参数设置的。

- `account-id:<federated-user-caller-specified-name>` 表示 Amazon STS 联合用户会话的唯一 ID。IAM 用户通过调用 `GetFederationToken` API 创建了此会话。如果为 Amazon STS 联合用户会话指定唯一 ID，这将拒绝已命名的联合会话，而不会影响已创建凭证的 IAM 用户的权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "aws:userId": [
            "AIDAXUSER1",
            "AROAXROLE1:*",
            "AROAXROLE2:<caller-specified-role-session-name>",
            "123456789012:<federated-user-caller-specified-name>"
          ]
        }
      }
    }
  ]
}
```

有关主体键值的具体示例，请参阅 主体键值。有关 IAM 唯一标识符以及如何获取此标识符的信息，请参阅唯一标识符。

拒绝访问使用基于资源的策略的指定主体

要限制访问使用基于资源的策略的特定主体，可以在 `Condition` 元素中使用条件上下文键 aws:PrincipalArn 或 aws:SourceIdentity。基于资源的策略是附加到资源的权限策略，可用于指定有权访问资源的用户，以及此类用户可以对资源执行的操作。

使用 aws:PrincipalARN 条件键时，您可以在策略的 Condition 元素中指定与临时安全凭证关联的 IAM 用户、角色或 Amazon STS 联合用户会话的 ARN。以下示例策略演示如何在基于资源的策略中使用 aws:PrincipalARN 上下文键：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Principal": "*",
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
    "Condition": {
      "ArnEquals": {
        "aws:PrincipalArn": [
          "arn:aws:iam::222222222222:role/ROLENAME",
          "arn:aws:iam::222222222222:user/USERNAME",
          "arn:aws:sts::222222222222:federated-user/USERNAME"
        ]
      }
    }
  }
}
```

使用 aws:SourceIdentity 上下文键时，需要在策略的 Condition 元素中指定与角色的临时安全凭证关联的源身份值。只要在主体使用任何 Amazon STS assume-role* CLI 命令或 Amazon STS AssumeRole* API 操作担任角色时通过设置 SourceIdentity 请求参数发出角色会话，此方法就适用。以下示例演示如何在基于资源的策略中使用 aws:SourceIdentity 上下文键：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Principal": "*",
    "Effect": "Deny",
    "Action": "s3:*",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
```

```
    "Condition": {
      "StringLike": {
        "aws:SourceIdentity": [
          "nikki_wolf@example.com",
          "<source identity value>"
        ]
      }
    }
  }
}
```

如果您仅更新主体的基于身份的策略，则用户仍可以执行基于资源的策略中允许的操作，但基于身份的策略中明确拒绝的操作除外。

拒绝访问基于资源的策略中的指定主体

1. 请参阅 使用 IAM 的 Amazon 服务 以查看该服务是否支持基于资源的策略。
2. 登录到 Amazon Web Services 管理控制台，然后打开服务控制台。每项服务在控制台中用于附加策略的位置都各不相同。
3. 编辑基于资源的策略。添加拒绝策略语句来指定凭证的识别信息：
   a. 在 `Principal` 元素中，输入通配符 (*)。主体将在 `Condition` 元素中受到限制。
   b. 在 `Effect` 元素中，输入“Deny”。
   c. 在 `Action` 中，输入服务命名空间和要拒绝的操作名称。要拒绝所有操作，请使用通配符（`*`）。例如：`"s3:*"`。
   d. 在 `Resource` 元素中，输入目标资源的 ARN。例如：`"arn:aws:s3:::amzn-s3-demo-bucket"`。
   e. 在 `Condition` 元素中，指定 `aws:PrincipalARN` 或 `aws:SourceIdentity` 上下文键。

      如果使用 `aws:PrincipalARN` 上下文键，输入要拒绝其访问的主体的 ARN

      如果使用 `aws:SourceIdentity` 上下文键，输入在角色会话中设置的源身份值以拒绝其访问。
4. 保存您的工作。

授予创建临时安全凭证的权限

默认情况下，IAM 用户无权为 Amazon STS 联合用户会话和角色创建临时安全凭证。您必须使用策略来向用户提供这些权限。虽然您可以直接向用户授予权限，但我们强烈建议您向组授予权限。这样可以使权限管理轻松得多。如果某个用户不再需要执行与权限关联的任务时，您只需从组中将其删除。如果其他用户需要执行这些任务，请将这些用户添加到组以授予权限。

要向 IAM 组授予为 Amazon STS 联合用户会话或角色创建临时安全凭证的权限，应附加一个策略，该策略授予以下一项或两项权限：

- 对于要访问 IAM 角色的 OIDC 和 SAML 联合主体，请授予对 Amazon STS `AssumeRole` 的访问权限。
- 对于无需角色的 Amazon STS 联合用户，请授予对 Amazon STS `GetFederationToken` 的访问权限。

有关 `AssumeRole` 和 `GetFederationToken` API 操作之间的差异的更多信息，请参阅请求临时安全凭证。

IAM 用户也可以调用 `GetSessionToken` 以创建临时安全凭证。用户无需任何权限即可调用 `GetSessionToken`。此操作旨在使用 MFA 验证用户身份。您不能使用策略来控制身份验证。这意味着，您不能阻止 IAM 用户调用 `GetSessionToken` 来创建临时凭证。

Example为授予担任角色的权限的示例策略

以下示例策略为 Amazon Web Services 账户 123123123123 中的 UpdateApp 角色授予调用 `AssumeRole` 的权限。在使用 `AssumeRole` 时，代表联合身份用户创建安全凭证的用户（或应用程序）无法委派尚未在角色权限策略中指定的任何权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:AssumeRole",
    "Resource": "arn:aws:iam::123123123123:role/UpdateAPP"
  }]
}
```

Example示例策略，该策略授予为联合身份用户创建临时安全凭证的权限

以下示例策略授予访问 GetFederationToken 的权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": "*"
  }]
}
```

> **Important**
>
> 使用 GetFederationToken 向 IAM 用户授予为 Amazon STS 联合用户创建临时安全凭证的权限时应注意，这将允许这些用户委派自己的权限。有关跨 IAM 用户和 Amazon Web Services 账户 委派权限的更多信息，请参阅 委派访问权限的策略示例。有关控制临时安全凭证权限的详细信息，请参阅临时安全凭证的权限。

Example向用户授予为联合身份用户创建临时安全凭证的有限权限的示例策略

在让 IAM 用户调用 GetFederationToken 时，最佳实践是限制 IAM 用户可以委派的权限。举例来说，以下策略展示如何让 IAM 用户仅为其名称以 Manager 开头的 Amazon STS 联合用户创建临时安全凭证。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Action": "sts:GetFederationToken",
    "Resource": ["arn:aws:sts::123456789012:federated-user/Manager*"]
  }]
```

```
}
```

授予使用身份增强控制台会话的权限

身份增强控制台会话允许在 Amazon IAM Identity Center 用户登录时将用户和会话 ID 包含在他们的 Amazon 控制台会话中。例如，Amazon Q Developer Pro 使用身份增强控制台会话来个性化服务体验。有关身份增强控制台会话的更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的 Enabling identity-enhanced console sessions。有关 Amazon Q 开发者版设置的信息，请参阅《Amazon Q 开发者版用户指南》中的设置 Amazon Q 开发者版。

要使身份增强控制台会话可供用户使用，您必须使用基于身份的策略向 IAM 主体授予对代表其控制台会话的资源的 `sts:SetContext` 权限。

> **Important**
>
> 默认情况下，用户无权为其身份增强控制台会话设置上下文。要允许这样做，您必须在基于身份的策略中向 IAM 主体授予 `sts:SetContext` 权限，如以下策略示例所示。

以下示例基于身份的策略向 IAM 主体授予 `sts:SetContext` 权限，允许主体为自己的 Amazon 控制台会话设置身份增强控制台会话上下文。策略资源 `arn:aws:sts::`*`account-id`*`:self` 代表调用方的 Amazon 会话。如果在多个账户中部署了相同的权限策略，例如使用 IAM Identity Center 权限集部署此策略，则可以将 `account-id` ARN 分段替换为通配符 `*`。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:SetContext",
            "Resource": "arn:aws:sts::111122223333:self"
        }
    ]
}
```

管理 Amazon Web Services 区域中的 Amazon STS

区域端点是 Amazon Web 服务在特定区域内入口点的 URL。Amazon 建议使用区域 Amazon Security Token Service (Amazon STS) 端点而不是全局端点，以减少延迟、建立冗余以及提高会话令牌的有效性。尽管全球（旧版）Amazon STS 端点 `https://sts.amazonaws.com` 高度可用，但它托管在单个 Amazon 区域美国东部（弗吉尼亚州北部），并且与其他端点一样，它不提供到其他区域中端点的自动故障转移。

- 减少延迟 - 通过向在地理位置离您较近的服务和应用程序的端点发出 Amazon STS 调用，可以缩短访问 Amazon STS 服务时的响应时间并减少延迟。
- 建立冗余 – 您可以将工作负载内故障的影响，限制在影响控制范围可预测的少数组件范围内。使用区域 Amazon STS 端点可以确保组件的范围与会话令牌的范围保持一致。有关此可靠性支柱的更多信息，请参阅在《Amazon Well-Architected Framework》中的 使用故障隔离来保护工作负载。
- 提高会话令牌有效性 – 来自区域 Amazon STS 端点的会话令牌在所有 Amazon Web Services 区域中都有效。来自全球 STS 端点的会话令牌仅在默认启用的 Amazon Web Services 区域 中有效。如果打算为您的账户启用新的区域，您可以使用来自区域 Amazon STS 端点的会话令牌。如果选择使用全球端点，您必须更改全球端点的 Amazon STS 会话令牌的区域兼容性。这样做可以确保令牌在所有 Amazon Web Services 区域 中都有效。

有关 Amazon STS 区域及其端点的列表，请参阅 Amazon STS 区域和端点。

> **Note**
>
> Amazon 已对默认启用的区域中的 Amazon Security Token Service (Amazon STS) 全局端点 (`https://sts.amazonaws.com`) 进行了更改，以增强其恢复能力和性能。发往全局端点的 Amazon STS 请求将自动在与您的工作负载相同的 Amazon Web Services 区域中处理。这些更改不会部署到选择加入的区域。我们建议您使用适当的 Amazon STS 区域端点。有关更多信息，请参阅 Amazon STS 全局端点更改。

主题

- 管理中国（北京）区域的 Amazon STS 端点会话令牌
- 在 Amazon Web Services 区域 中激活和停用 Amazon STS
- 编写代码以使用 Amazon STS 区域

管理中国（北京）区域的 Amazon STS 端点会话令牌

如果您账户中的工作负载或调用方打算使用签名版本 4A（Sigv4a）算法签署 Amazon 请求，或者您计划采用需要它的特定 Amazon Web Services 服务 功能，则需要将中国（北京）区域的 Amazon STS 端点配置为提供支持 Sigv4A 的会话令牌。默认情况下，中国（北京）区域的 Amazon STS 端点提供仅支持签名版本 4（SigV4）算法的会话令牌。支持 Sigv4a 算法的会话令牌比支持 Sigv4 算法的会话令牌大，并且与中国（宁夏）区域的 Amazon STS 端点发行的会话令牌大小匹配，后者已支持使用 Sigv4A。更改该设置可能会影响临时存储令牌的现有系统。

您可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 更改该设置。

更改中国（北京）区域 STS 端点的会话令牌的区域兼容性（控制台）

1. 以有权执行 IAM 管理任务的用户身份登录。要更改会话令牌的兼容性，您必须具有允许 `iam:SetSecurityTokenServicePreferences` 操作的策略。
2. 打开 IAM 控制台。在导航窗格中，选择 Account settings（账户设置）。
3. 在 Security Token Service (STS) 部分的 Session Tokens from the STS endpoints（来自 STS 端点的会话令牌）。Amazon Web Services 中国（北京）区域端点表示 `Support only SigV4 based signatures on Amazon Web Services requests`。选择 Change (更改)。
4. 在更改区域兼容性对话框中，选择 SigV4 和 SigV4A 算法。然后选择 Save changes （保存更改)。

> **Note**
>
> 与基于签名版本 4A（SigV4A）的签名兼容的会话令牌包含的字符数多于与基于签名版本 4（SigV4）的签名兼容的令牌。更改该设置可能会影响临时存储令牌的现有系统。

更改中国（北京）区域 STS 端点的会话令牌的区域兼容性（Amazon CLI）

设置会话令牌版本。版本 1 令牌仅与基于 Sigv4 的签名兼容。版本 2 令牌与基于 Sigv4 和 SigV4A 的签名兼容。不过，版本 2 令牌包含更多字符，可能会影响临时存储令牌的系统。

- `aws iam set-security-token-service-preferences`

更改中国（北京）区域 STS 端点的会话令牌的区域兼容性（Amazon API）

设置会话令牌版本。版本 1 令牌仅与基于 Sigv4 的签名兼容。版本 2 令牌与基于 Sigv4 和 SigV4A 的签名兼容。不过，版本 2 令牌包含更多字符，可能会影响临时存储令牌的系统。

- SetSecurityTokenServicePreferences

在 Amazon Web Services 区域 中激活和停用 Amazon STS

在为区域激活 Amazon STS 端点时，Amazon STS 可能会向您的账户中发出 Amazon STS 请求的用户和角色颁发临时凭证。然后，可以在默认启用或手动启用的任何区域中使用这些凭证。对于默认启用的区域，您必须在生成临时凭证的账户中激活该区域 Amazon STS 端点。在发出请求时，用户是登录到同一账户还是不同账户并不重要。当使用手动启用的区域为另一个 Amazon Web Services 账户 中的角色请求临时凭证时，目标账户（包含该角色的账户）必须为该区域启用 Amazon STS 操作。此举可确保临时安全凭证能够正确生成。

例如，假设账户 A 中的用户要向 Amazon STS 区域端点 `https://sts.ap-southeast-3.amazonaws.com` 发送 `sts:AssumeRole` API 请求。该请求旨在为账户 B 中名为 `Developer` 的角色获取临时凭证。由于该请求旨在为账户 B 中的实体创建凭证，因此，账户 B 必须激活 `ap-southeast-3` 区域。账户 A（或任何其他账户）中的用户可以调用 `ap-southeast-3` Amazon STS 端点，以便为账户 B 请求凭证，而无论是否在其账户中激活了该区域。欲了解更多信息，请参阅在您的账户中启用或禁用 Amazon Web Services 区域。

> **Note**
>
> 活动区域适用于该账户中使用临时凭证的每个用户。要控制哪些 IAM 用户或角色可以访问区域，请在权限策略中使用 `aws:RequestedRegion` 条件键。

在默认启用的区域中激活或停用 Amazon STS（控制台）

1. 以根用户或有权执行 IAM 管理任务的用户身份登录。
2. 打开 IAM 控制台，然后在导航窗格中选择 Account settings（账户设置）。
3. 在 Security Token Service (STS) 部分的 Endpoints（端点）中，找到要配置的区域，然后在 STS status（STS 状态）列中选择 Active（活动）或 Inactive（非活动）。
4. 在打开的对话框中，选择 Activate（激活）或 Deactivate（停用）。

对于必须启用的区域，当您启用相关区域时，我们会自动激活 Amazon STS。启用区域后，Amazon STS 对于该区域将始终处于活动状态，您无法将其停用。要了解如何启用默认禁用的区域，请参阅《Amazon 账户管理 参考指南》中的指定您的账户可以使用的 Amazon Web Services 区域。

编写代码以使用 Amazon STS 区域

在激活区域后，您可以将 Amazon STS API 调用定向到该区域。以下 Java 代码段说明了如何配置 `AWSSecurityTokenService` 对象，以向欧洲地区（米兰）（eu-south-1）区域发出请求。

```
EndpointConfiguration regionEndpointConfig = new EndpointConfiguration("https://sts.eu-
south-1.amazonaws.com", "eu-south-1");
AWSSecurityTokenService stsRegionalClient =
 AWSSecurityTokenServiceClientBuilder.standard()
.withCredentials(credentials)
.withEndpointConfiguration(regionEndpointConfig)
.build();
```

Amazon STS 建议您调用区域端点。要了解如何手动启用的区域，请参阅《Amazon 账户管理 Reference Guide》中的 Specify which Amazon Web Services 区域 your account can use。

在该示例中，第一行将名为 `regionEndpointConfig` 的 `EndpointConfiguration` 对象实例化，并将端点的 URL 和 Amazon Web Services 区域 作为参数传递。

要了解如何使用 Amazon SDK 的环境变量设置 Amazon STS 区域端点，请参阅 Amazon SDK 和工具参考指南中的 Amazon STS 区域化端点。

有关所有其他的语言和编程环境组合，请参阅相关开发工具包的文档。

Amazon STS 区域和端点

> **Note**
>
> Amazon 已对默认启用的区域中的 Amazon Security Token Service (Amazon STS) 全局端点 (`https://sts.amazonaws.com`) 进行了更改，以增强其恢复能力和性能。发往全局端点的 Amazon STS 请求将自动在与您的工作负载相同的 Amazon Web Services 区域中处理。这些更改不会部署到选择加入的区域。我们建议您使用适当的 Amazon STS 区域端点。有关更多信息，请参阅 Amazon STS 全局端点更改。

下表列出了区域及其终端节点。其中指示哪些是默认激活的，哪些是可激活或停用的。

区域名称	端点	默认情况下激活	手动激活/停用
--全球--	sts.amazonaws.com	✓ 是	✗ 不支持
美国东部（俄亥俄州）	sts.us-east-2.amazonaws.com	✓ 是	✓ 是
美国东部（弗吉尼亚州北部）	sts.us-east-1.amazonaws.com	✓ 是	✗ 不支持
美国西部（北加利福尼亚）	sts.us-west-1.amazonaws.com	✓ 是	✓ 是
美国西部（俄勒冈州）	sts.us-west-2.amazonaws.com	✓ 是	✓ 是
非洲（开普敦）	sts.af-south-1.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（香港）	sts.ap-east-1.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（海得拉巴）	sts.ap-south-2.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（雅加达）	sts.ap-southeast-3.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（马来西亚）	sts.ap-southeast-5.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（墨尔本）	sts.ap-southeast-4.amazonaws.com	✗ 否[1]	✗ 不支持
亚太地区（孟买）	sts.ap-south-1.amazonaws.com	✓ 是	✓ 是
亚太地区（大阪）	sts.ap-northeast-3.amazonaws.com	✓ 是	✓ 是
亚太地区（首尔）	sts.ap-northeast-2.amazonaws.com	✓ 是	✓ 是
亚太地区（新加坡）	sts.ap-southeast-1.amazonaws.com	✓ 是	✓ 是
亚太地区（悉尼）	sts.ap-southeast-2.amazonaws.com	✓ 是	✓ 是

区域名称	端点	默认情况下激活	手动激活/停用
亚太地区（泰国）	sts.ap-southeast-7.amazonaws.com	⊗ 否[1]	⊗ 不支持
亚太地区（东京）	sts.ap-northeast-1.amazonaws.com	⊘ 是	⊘ 是
加拿大（中部）	sts.ca-central-1.amazonaws.com	⊘ 是	⊘ 是
加拿大西部（卡尔加里）	sts.ca-west-1.amazonaws.com	⊗ 否[1]	⊗ 不支持
中国（北京）	sts.cn-north-1.amazonaws.com.cn	⊘ 是[2]	⊗否
中国（宁夏）	sts.cn-northwest-1.amazonaws.com.cn	⊘ 是[2]	⊘ 是
欧洲地区（法兰克福）	sts.eu-central-1.amazonaws.com	⊘ 是	⊘ 是
欧洲地区（爱尔兰）	sts.eu-west-1.amazonaws.com	⊘ 是	⊘ 是
欧洲地区（伦敦）	sts.eu-west-2.amazonaws.com	⊘ 是	⊘ 是
欧洲地区（米兰）	sts.eu-south-1.amazonaws.com	⊗ 否[1]	⊗ 不支持
欧洲地区（巴黎）	sts.eu-west-3.amazonaws.com	⊘ 是	⊘ 是
欧洲（西班牙）	sts.eu-south-2.amazonaws.com	⊗ 否[1]	⊗ 不支持
欧洲地区（斯德哥尔摩）	sts.eu-north-1.amazonaws.com	⊘ 是	⊘ 是
欧洲（苏黎世）	sts.eu-central-2.amazonaws.com	⊗ 否[1]	⊗ 不支持
以色列（特拉维夫）	sts.il-central-1.amazonaws.com	⊗ 否[1]	⊗ 不支持
墨西哥（中部）	sts.mx-central-1.amazonaws.com	⊗ 否[1]	⊗ 不支持
中东（巴林）	sts.me-south-1.amazonaws.com	⊗ 否[1]	⊗ 不支持
中东（阿联酋）：	sts.me-central-1.amazonaws.com	⊗ 否[1]	⊗ 不支持

区域名称	端点	默认情况下激活	手动激活/停用
南美洲（圣保罗）	sts.sa-east-1.amazonaws.com	✓ 是	✓ 是

您必须启用区域才能使用它。这会自动激活 Amazon STS。您无法在这些区域中手动激活或停用 Amazon STS。

[2]要在中国使用 Amazon，您需要中国 Amazon 的特定账户和凭证。

Amazon STS 全局端点更改

Amazon 已对默认启用的区域中的 Amazon Security Token Service (Amazon STS) 全局端点 (`https://sts.amazonaws.com`) 进行了更改，以增强其恢复能力和性能。以前，发往 Amazon STS 全局端点的所有请求都由 Amazon Web Services 区域美国东部（弗吉尼亚州北部）处理。现在，在默认启用的区域中，发往 Amazon STS 全局端点的请求将自动在请求发起的同一区域（而非美国东部（弗吉尼亚州北部）区域）中处理。这些更改不会部署到选择加入的区域。

通过此更改，Amazon STS 将根据使用的原始区域和 DNS 解析程序处理您的请求。如果 Amazon STS 全局端点的 DNS 请求由默认启用的区域中的 Amazon DNS 服务器处理，则发往 Amazon 全局端点的请求将在与您部署的 Amazon STS 工作负载相同的区域中处理。如果您的请求源自选择加入区域，或者使用 Amazon DNS 服务器以外的 DNS 解析程序解析您的请求，则发往 Amazon STS 全局端点的请求将继续在美国东部（弗吉尼亚州北部）区域处理。有关 Amazon DNS 的更多信息，请参阅《Amazon Virtual Private Cloud 用户指南》中的 Amazon DNS 服务器。

下表显示了如何根据您的 DNS 提供商路由发往 Amazon STS 全局端点的请求。

DNS 解析程序	发往 Amazon STS 全局端点的请求是否路由到本地 Amazon Web Services 区域？
默认启用的区域中的 Amazon VPC 中的 Amazon DNS 解析程序	是
选择加入区域中的 Amazon VPC 中的 Amazon DNS 解析程序	不，请求将路由到美国东部（弗吉尼亚州北部）区域
由您的 ISP、公共 DNS 提供商或任何其他 DNS 提供商提供的 DNS 解析程序	不，请求将路由到美国东部（弗吉尼亚州北部）区域

为了确保最大限度地减少对现有流程的干扰，Amazon 已实施以下措施：

- 对 Amazon STS 全局端点发出的请求的 Amazon CloudTrail 日志将发送到美国东部（弗吉尼亚州北部）区域。由 Amazon STS 区域端点处理的请求的 CloudTrail 日志将继续记录到 CloudTrail 中的相应区域。
- Amazon STS 全局端点和区域端点执行的操作的 CloudTrail 日志将包含附加字段 `endpointType` 和 `awsServingRegion`，以指示哪个端点和区域处理了请求。有关 CloudTrail 日志示例，请参阅CloudTrail 日志文件中使用全局端点的示例 Amazon STS 事件。
- 无论哪个区域处理请求，发往 Amazon STS 全局端点的请求的 `aws:RequestedRegion` 条件键的值都将为 `us-east-1`。
- Amazon STS 全局端点处理的请求不会与区域 Amazon STS 端点共享每秒请求数配额。

如果您在选择加入区域中拥有工作负载并且仍在使用 Amazon STS 全局端点，我们建议您迁移到 Amazon STS 区域端点以提高恢复能力和性能。有关配置区域 Amazon STS 端点的更多信息，请参阅《Amazon SDK 和工具参考指南》中的 Amazon STS Regional endpoints。

Amazon CloudTrail 和区域端点

对区域性和全球性端点的调用将在 Amazon CloudTrail 的 `tlsDetails` 字段中记录。对区域性端点（例如 `us-east-2.amazonaws.com`）的调用将在 CloudTrail 中记录到相应的区域。对全球终端节点 (如 `sts.amazonaws.com`) 的调用会记录为对全球服务的调用。全球性 Amazon STS 端点的事件将记录到 us-east-1 区域。

> **Note**
>
> 只能对支持此字段的服务查看 `tlsDetails`。请参阅《Amazon CloudTrail 用户指南》中的在 CloudTrail 中支持 TLS 详细信息的服务
> 有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。

使自定义身份凭证代理程序能够访问 Amazon 控制台

您可以通过编写和运行代码来创建 URL 以使登录到您组织网络的用户能够安全访问 Amazon Web Services 管理控制台。该 URL 包含您从 Amazon 获得的登录令牌，而令牌则用于对访问 Amazon 的用户进行身份验证。由于联合身份验证的原因，显示的控制台会话可能包含不同的 `AccessKeyId`。要通过相关 CloudTrail 事件跟踪联合身份验证登录的访问密钥使用情况，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用 和 Amazon Web Services 管理控制台 登录事件。

> **Note**
>
> 如果您的组织使用与 SAML 兼容的身份提供者 (IdP)，则无需编写代码即可设置对控制台的访问权限。这适用于 Microsoft 的 Active Directory 联合身份验证服务或开源 Shibboleth 等提供商。有关更多信息，请参阅 使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。

要使您组织的用户能够访问 Amazon Web Services 管理控制台，可以创建执行以下步骤的自定义身份凭证代理程序：

1. 确认您的本地身份系统已对用户进行身份验证。
2. 调用 Amazon Security Token Service (Amazon STS) AssumeRole（推荐）或 GetFederationToken API 操作为该用户获取临时安全凭证。要了解在担任角色时使用的各种方法，请参阅担任角色的方法。要了解在获取安全凭证时如何传递可选的会话标签，请参阅在 Amazon STS 中传递会话标签。
   - 如果使用某个 `AssumeRole*` API 操作获取角色的临时安全凭证，您可以在调用中包含 `DurationSeconds` 参数。该参数指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的角色会话持续时间。在 `AssumeRole*` 操作中使用 `DurationSeconds` 时，必须以具有长期凭证的 IAM 用户身份调用该操作。否则，在步骤 3 中对联合端点的调用将失败。要了解如何查看或更改角色的最大值，请参阅更新角色的最长会话持续时间。
   - 如果使用 `GetFederationToken` API 操作获取凭证，您可以在调用中包含 `DurationSeconds` 参数。该参数指定您的角色会话的持续时间。该值的范围是 900 秒 (15 分钟) 到 129,600 秒 (36 小时)。您只能使用 IAM 用户的长期 Amazon 安全凭证进行该 API 调用。您还可以使用 Amazon Web Services 账户根用户凭证进行这些调用，但我们不建议使用这种方法。如果您以根用户身份进行该调用，则会话默认持续 1 小时。或者，您可以指定 900 秒 (15 分钟) 到 3,600 秒 (1 小时) 之间的会话。
3. 调用 Amazon 联合端点并提供临时安全凭证来请求登录令牌。
4. 构造包含该令牌的控制台 URL：
   - 如果在 URL 中使用某个 `AssumeRole*` API 操作，您可以包含 `SessionDuration` HTTP 参数。该参数指定控制台会话持续时间，范围是 900 秒 (15 分钟) 到 43200 秒 (12 小时)。
   - 如果在 URL 中使用 `GetFederationToken` API 操作，您可以包含 `DurationSeconds` 参数。该参数指定联合控制台会话的持续时间。该值的范围是 900 秒 (15 分钟) 到 129,600 秒 (36 小时)。

> **Note**
>
> - 您的 `SessionDuration` 不能大于或等于您所代入的角色的最大会话持续时间设置。例如，您将要代入角色的最大会话持续时间设置为 5 小时。您的 `SessionDuration` 参数可以是 16524 秒或 4 小时 59 秒。
> - 使用 `GetFederationToken` 获取临时凭证时，请不要使用 `SessionDuration` HTTP 参数。该操作将失败。
> - 使用一个角色的凭证担任其他角色称为 role chaining（角色链）。在使用角色链时，新凭证的最大持续时间限制为 1 小时。当您使用角色向 EC2 实例上运行的应用程序授予权限时，则这些应用程序不受制于此限制。
> - 通过角色串联获取临时凭证时，请不要使用 `SessionDuration` HTTP 参数。该操作将失败。

5. 将 URL 分配给用户或代表用户调用 URL。

联合端点提供的 URL 在创建后的 15 分钟内有效。这不同于与 URL 关联的临时安全凭证会话的持续时间 (以秒为单位)。这些凭证在创建时指定的持续时间内有效，从创建时算起。

> **Important**
>
> 如果您在关联的临时安全凭证中启用了权限，则该 URL 授予通过 Amazon Web Services 管理控制台访问您的 Amazon 资源的权限。因此，您应该视 URL 为机密。我们建议您通过安全的重定向返回 URL，例如，在 SSL 连接上使用 302 HTTP 响应状态代码。有关 302 HTTP 响应状态代码的详细信息，请参阅 RFC 2616，第 10.3.3 节。

要完成这些任务，您可以使用适用于 Amazon Identity and Access Management 的 HTTPS 查询 API (IAM) 和 Amazon Security Token Service (Amazon STS)。或者，您可以将编程语言（例如 Java、Ruby 或 C#）与相应的 Amazon 开发工具包结合使用。以下主题分别说明了这些方式。

主题

- 使用 IAM 查询 API 操作的示例代码
- 使用 Python 的示例代码
- 使用 Java 的示例代码
- 演示如何构造 URL 的示例 (Ruby)

使用 IAM 查询 API 操作的示例代码

您可以构造一个 URL，此 URL 向角色和联合身主体授予对 Amazon Web Services 管理控制台的直接访问权限。此任务使用 IAM 和 Amazon STS HTTPS 查询 API。有关提出查询请求的详细信息，请参阅提出查询请求。

> **Note**
>
> 以下过程包括了文本字符串的例子。为增加可读性，一些较长的例子中添加了换行符。如果您要创建并使用这些字符串，必须省略所有换行符。

向角色和联合主体授予从 Amazon Web Services 管理控制台访问您的资源的权限

1. 在您的身份验证系统里验证该用户
2. 为用户获取临时安全凭证。临时凭证由访问密钥 ID、秘密访问密钥和会话令牌组成。有关创建临时凭证的详细信息，请参阅IAM 临时安全凭证。

   要获取临时凭证，您可以调用 Amazon STS AssumeRole API（推荐）或 GetFederationToken API。有关这些 API 操作之间区别的更多信息，请参阅 Amazon 安全博客中的了解安全地委派您的 Amazon 账户访问权限的 API 选项。

   > **Important**
   >
   > 在使用 GetFederationToken API 创建临时安全凭证时，您必须指定这些凭证为担任角色的用户授予的权限。对于任何以 `AssumeRole*` 开头的 API 操作，可使用 IAM 角色来分配权限。对于其他 API 操作，机制因 API 而异。有关更多详细信息，请参阅临时安全凭证的权限。此外，如果使用 `AssumeRole*` API 操作，您还必须以具有长期凭证的 IAM 用户身份调用这些操作。否则，在步骤 3 中对联合端点的调用将失败。

3. 在获取临时安全凭证后，将其构建到 JSON 会话字符串以将其交换为登录令牌。下例展示如何将凭证编码。请将占位符文本替换为上一步骤中接收的凭证中的相应值。

   ```
   {"sessionId":"*** temporary access key ID ***",
   "sessionKey":"*** temporary secret access key ***",
   "sessionToken":"*** session token ***"}
   ```

4. URL 编码上一步骤中的会话字符串。由于您编码的信息是敏感信息，因此建议您避免对此编码使用 Web 服务。请改用开发工具包中在本地安装的函数或功能来安全地编码这些信息。您可以

使用 Python 的 `urllib.quote_plus` 函数、Java 的 `URLEncoder.encode` 函数或 Ruby 的 `CGI.escape` 函数。请参阅本主题后面的示例。

5.

> **Note**
>
> Amazon 在此处支持 POST 请求。

将您的请求发送到 Amazon 联合身份验证端点：

`https://`*`rergion-code`*`signin.amazonaws.cn/federation`

有关可能的 *`region-code`* 值的列表，请参阅 Amazon 登录端点 中的 Region（区域）列。您可以选择使用默认 Amazon 登录联合身份验证端点：

`https://signin.amazonaws.cn/federation`

该请求必须包含 `Action` 和 `Session` 参数；(可选) 如果使用了 `AssumeRole*` API 操作，还必须包含 `SessionDuration` HTTP 参数，如以下示例中所示。

```
Action = getSigninToken
SessionDuration = time in seconds
Session = *** the URL encoded JSON string created in steps 3 & 4 ***
```

> **Note**
>
> 此步骤中的以下说明仅适用于 GET 请求。

`SessionDuration` HTTP 参数指定控制台会话的持续时间。它不同于使用 `DurationSeconds` 参数指定的临时凭证的持续时间。可以将 `SessionDuration` 的最大值指定为 43200（12 小时）。如果缺少 `SessionDuration` 参数，则会话默认为在步骤 2 中从 Amazon STS 检索的凭证的持续时间（默认为 1 小时）。有关如何使用 `DurationSeconds` 参数指定持续时间的详细信息，请参阅 `AssumeRole` API 的文档。创建超过 1 小时的控制台会话的功能是联合端点的 `getSigninToken` 操作所固有的。

> **Note**
>
> - 您的 SessionDuration 不能大于或等于您所代入的角色的最大会话持续时间设置。例如，您将要代入角色的最大会话持续时间设置为 5 小时。您的 SessionDuration 参数可以是 16524 秒或 4 小时 59 秒。
> - 使用 GetFederationToken 获取临时凭证时，请不要使用 SessionDuration HTTP 参数。该操作将失败。
> - 使用一个角色的凭证担任其他角色称为 role chaining（角色链）。在使用角色链时，新凭证的最大持续时间限制为 1 小时。当您使用角色向 EC2 实例上运行的应用程序授予权限时，则这些应用程序不受制于此限制。
> - 通过角色串联获取临时凭证时，请不要使用 SessionDuration HTTP 参数。该操作将失败。

在启用具有更长持续时间的控制台会话时，您增加了凭证外泄的风险。为了帮助缓解这种风险，您可以通过在 IAM 控制台页面的 Role Summary（角色摘要）上选择 Revoke Sessions（撤消会话）立即禁用所有角色的有效控制台会话。有关更多信息，请参阅 撤销 IAM 角色临时安全凭证。

以下为请求具体形式的示例。在此处换行以便阅读，但您应将其作为一行字符串提交。

```
https://signin.amazonaws.cn/federation
?Action=getSigninToken
&SessionDuration=1800
&Session=%7B%22sessionId%22%3A+%22ASIAJUMHIZPTOKTBMK5A%22%2C+%22sessionKey%22
%3A+%22LSD7LWI%2FL%2FN%2BgYpan5QFz0XUpc8s7HYjRsgcsrsm%22%2C+%22sessionToken%2
2%3A+%22FQoDYXdzEBQaDLbj3VWv2u50NN%2F3yyLSASwYtWhPnGPMNmzZFfZsL0Qd3vtYHw5A5dW
AjOsrkdPkghomIe3mJip5%2F0djDBbo7SmO%2FENDEiCdpsQKodTpleKA8xQq0CwFg6a69xdEBQT8
FipATnLbKoyS4b%2FebhnsTUjZZQWp0wXXqFF7gSm%2FMe2tXe0jzsdP0O12obez9lijPSdF1k2b5
PfGhiuyAR9aD5%2BubM0pY86fKex1qsytjvyTbZ9nXe6DvxVDcnCOhOGETJ7XFkSFdH0v%2FYR25C
UAhJ3nXIkIbG7Ucv9cOEpCf%2Fg23ijRgILIBQ%3D%3D%22%7D
```

来自联合端点的响应是一个具有 SigninToken 值的 JSON 文档。类似下例。

```
{"SigninToken":"*** the SigninToken string ***"}
```

6.

> **Note**
>
> Amazon 在此处支持 POST 请求。

最后，创建您用户可用于访问 Amazon Web Services 管理控制台的 URL。此 URL 与您在Step 5 中使用的联合 URL 端点相同，外加以下参数：

```
?Action = login
&Issuer = *** the form-urlencoded URL for your internal sign-in page ***
&Destination = *** the form-urlencoded URL to the desired Amazon console page ***
&SigninToken = *** the value of SigninToken received in the previous step ***
```

> **Note**
>
> 此步骤中的以下说明仅适用于 GET API。

下列 URL 为 URL 最终形式的示例。URL 的有效期为 15 分钟，自创建时算起。在 URL 中嵌入的临时安全凭证和控制台会话的有效期为在最初请求它们时在 SessionDuration HTTP 参数中指定的持续时间。

```
https://signin.amazonaws.cn/federation
?Action=login
&Issuer=http%3A%2F%2Fexample.com
&Destination=http%3A%2F%2Fconsole.aws.amazon.com%2F
&SigninToken=VCQgs5qZZt3Q6fn8Tr5EXAMPLEmLnwB7JjUc-SHwnUUWabcRdnWsi4DBn-dvC
CZ85wrD0nmldUcZEXAMPLE-vXYH4Q__mleuF_W2BE5HYexbe9y4Of-kje53SsjNNecATfjIzpW1
WibbnH6YcYRiBoffZBGExbEXAMPLE5aiKX4THWjQKC6gg6alHu6JFrnOJoK3dtP6I9a6hi6yPgm
iOkPZMmNGmhsvVxetKzr8mx3pxhHbMEXAMPLETv1pij0rok3IyCR2YVcIjqwfWv32HU2Xlj471u
3fU6uOfUComeKiqTGX974xzJOZbdmX_t_lLrhEXAMPLEDDIisSnyHGw2xaZZqudm4mo2uTDk9Pv
9l5K0ZCqIgEXAMPLEcA6tgLPykEWGUyH6BdSC6166n4M4JkXIQgac7_7821YqixsNxZ6rsrpzwf
nQoS14O7R0eJCCJ684EXAMPLEZRdBNnuLbUYpz2Iw3vIN0tQgOujwnwydPscM9F7foaEK3jwMkg
Apeb1-6L_OB12MZhuFxx55555EXAMPLEhyETEd4ZulKPdXHkgl6T9ZkIlHz2Uy1RUTUhhUxNtSQ
nWc5xkbBoEcXqpoSIeK7yhje9Vzhd61AEXAMPLElbWeouACEMG6-Vd3dAgFYd6i5FYoyFrZLWvm
0LSG7RyYKeYN5VIzUk3YWQpyjP0RiT5KUrsUi-NEXAMPLExMOMdoODBEgKQsk-iu2ozh6r8bxwC
RNhujg
```

使用 Python 的示例代码

以下示例说明了如何使用 Python 以编程方式构造授予用户 Amazon Web Services 管理控制台直接访问权限的 URL。以下是两个示例：

- 通过 GET 请求联合到 Amazon
- 通过 POST 请求联合到 Amazon

这两个示例均使用 适用于 Python (Boto3) 的 Amazon SDK 和 AssumeRole API 获取临时安全凭证。

如果您的 AssumeRoleSession 凭证来自角色串联，请不要将 SessionDuration 包括在内。如果包括 SessionDuration，则操作将失败。

使用 GET 请求

使用 POST 请求

使用 Java 的示例代码

以下示例说明了如何使用 Java 以编程方式构造授予用户 Amazon Web Services 管理控制台直接访问权限的 URL。下列代码段使用 Amazon SDK for Java。

```
import java.net.URLEncoder;
import java.net.URL;
import java.net.URLConnection;
import java.io.BufferedReader;
import java.io.InputStreamReader;
// Available at http://www.json.org/java/index.html
import org.json.JSONObject;
import com.amazonaws.auth.AWSCredentials;
import com.amazonaws.auth.BasicAWSCredentials;
import com.amazonaws.services.securitytoken.AWSSecurityTokenServiceClient;
import com.amazonaws.services.securitytoken.model.Credentials;
import com.amazonaws.services.securitytoken.model.GetFederationTokenRequest;
import com.amazonaws.services.securitytoken.model.GetFederationTokenResult;


/* Calls to Amazon STS API operations must be signed using the access key ID
   and secret access key of an IAM user or using existing temporary
   credentials. The credentials should not be embedded in code. For
   this example, the code looks for the credentials in a
   standard configuration file.
```

```
*/
AWSCredentials credentials =
  new PropertiesCredentials(
         AwsConsoleApp.class.getResourceAsStream("AwsCredentials.properties"));

AWSSecurityTokenServiceClient stsClient =
  new AWSSecurityTokenServiceClient(credentials);

GetFederationTokenRequest getFederationTokenRequest =
  new GetFederationTokenRequest();
getFederationTokenRequest.setDurationSeconds(1800);
getFederationTokenRequest.setName("UserName");

// A sample policy for accessing Amazon Simple Notification Service (Amazon SNS) in the
 console.

String policy = "{\"Version\":\"2012-10-17\",       \"Statement\":[{\"Action\":\"sns:*
\"," +
  "\"Effect\":\"Allow\",\"Resource\":\"*\"}]}";

getFederationTokenRequest.setPolicy(policy);

GetFederationTokenResult federationTokenResult =
  stsClient.getFederationToken(getFederationTokenRequest);

Credentials federatedCredentials = federationTokenResult.getCredentials();

// The issuer parameter specifies your internal sign-in
// page, for example http://mysignin.internal.mycompany.com/.
// The console parameter specifies the URL to the destination console of the
// Amazon Management Console. This example goes to Amazon SNS.
// The signin parameter is the URL to send the request to.

String issuerURL = "http://mysignin.internal.mycompany.com/";
String consoleURL = "http://console.aws.amazon.com/sns";
String signInURL = "https://signin.amazonaws.cn/federation";

// Create the sign-in token using temporary credentials,
// including the access key ID,  secret access key, and session token.
String sessionJson = String.format(
  "{\"%1$s\":\"%2$s\",\"%3$s\":\"%4$s\",\"%5$s\":\"%6$s\"}",
  "sessionId", federatedCredentials.getAccessKeyId(),
  "sessionKey", federatedCredentials.getSecretAccessKey(),
  "sessionToken", federatedCredentials.getSessionToken());
```

```
// Construct the sign-in request with the request sign-in token action, a
// 12-hour console session duration, and the JSON document with temporary
// credentials as parameters.

String getSigninTokenURL = signInURL +
                           "?Action=getSigninToken" +
                           "&DurationSeconds=43200" +
                           "&SessionType=json&Session=" +
                           URLEncoder.encode(sessionJson,"UTF-8");

URL url = new URL(getSigninTokenURL);

// Send the request to the Amazon federation endpoint to get the sign-in token
URLConnection conn = url.openConnection ();

BufferedReader bufferReader = new BufferedReader(new
  InputStreamReader(conn.getInputStream()));
String returnContent = bufferReader.readLine();

String signinToken = new JSONObject(returnContent).getString("SigninToken");

String signinTokenParameter = "&SigninToken=" + URLEncoder.encode(signinToken,"UTF-8");

// The issuer parameter is optional, but recommended. Use it to direct users
// to your sign-in page when their session expires.

String issuerParameter = "&Issuer=" + URLEncoder.encode(issuerURL, "UTF-8");

// Finally, present the completed URL for the Amazon console session to the user

String destinationParameter = "&Destination=" + URLEncoder.encode(consoleURL,"UTF-8");
String loginURL = signInURL + "?Action=login" +
                     signinTokenParameter + issuerParameter + destinationParameter;
```

演示如何构造 URL 的示例 (Ruby)

以下示例说明了如何使用 Ruby 以编程方式构造授予用户 Amazon Web Services 管理控制台直接访问权限的 URL。下列代码段使用 Amazon SDK for Ruby。

```
require 'rubygems'
require 'json'
require 'open-uri'
```

```
require 'cgi'
require 'aws-sdk'

# Create a new STS instance
#
# Note: Calls to Amazon STS API operations must be signed using an access key ID
# and secret access key. The credentials can be in EC2 instance metadata
# or in environment variables and will be automatically discovered by
# the default credentials provider in the Amazon Ruby SDK.
sts = Aws::STS::Client.new()

# The following call creates a temporary session that returns
# temporary security credentials and a session token.
# The policy grants permissions to work
# in the Amazon SNS console.

session = sts.get_federation_token({
  duration_seconds: 1800,
  name: "UserName",
  policy: "{\"Version\":\"2012-10-17\",       \"Statement\":{\"Effect\":\"Allow\",
\"Action\":\"sns:*\",\"Resource\":\"*\"}}",
})

# The issuer value is the URL where users are directed (such as
# to your internal sign-in page) when their session expires.
#
# The console value specifies the URL to the destination console.
# This example goes to the Amazon SNS console.
#
# The sign-in value is the URL of the Amazon STS federation endpoint.
issuer_url = "http://mysignin.internal.mycompany.com/"
console_url = "http://console.aws.amazon.com/sns"
signin_url = "https://signin.amazonaws.cn/federation"

# Create a block of JSON that contains the temporary credentials
# (including the access key ID, secret access key, and session token).
session_json = {
  :sessionId => session.credentials[:access_key_id],
  :sessionKey => session.credentials[:secret_access_key],
  :sessionToken => session.credentials[:session_token]
}.to_json

# Call the federation endpoint, passing the parameters
# created earlier and the session information as a JSON block.
```

```
# The request returns a sign-in token that's valid for 15 minutes.
# Signing in to the console with the token creates a session
# that is valid for 12 hours.
get_signin_token_url = signin_url +
                       "?Action=getSigninToken" +
                       "&SessionType=json&Session=" +
                       CGI.escape(session_json)

returned_content = URI.parse(get_signin_token_url).read

# Extract the sign-in token from the information returned
# by the federation endpoint.
signin_token = JSON.parse(returned_content)['SigninToken']
signin_token_param = "&SigninToken=" + CGI.escape(signin_token)

# Create the URL to give to the user, which includes the
# sign-in token and the URL of the console to open.
# The "issuer" parameter is optional but recommended.
issuer_param = "&Issuer=" + CGI.escape(issuer_url)
destination_param = "&Destination=" + CGI.escape(console_url)
login_url = signin_url + "?Action=login" + signin_token_param +
  issuer_param + destination_param
```

Amazon Identity and Access Management 资源的标签

标签是您分配给 Amazon 资源的自定义属性标签。每个标签具有两个部分：

- 标签键（例如，CostCenter、Environment、Project 或 Purpose）。
- 一个称为标签值的可选字段（例如，111122223333、Production 或团队名称）。省略标签值与使用空字符串效果相同。

这些被统称为键-值对。有关 IAM 资源上可以拥有的标签数量的限制，请参阅 IAM 和 Amazon STS 配额。

> **Note**
>
> 有关标签键和标签键值区分大小写的详细信息，请参阅 Case sensitivity。

标签有助于您标识和组织 Amazon 资源。许多 Amazon 服务支持标记，因此，您可以将同一标签分配给来不同服务的资源，以指示这些资源是相关的。例如，您可以将相同的标签分配给为 Amazon S3 存储桶分配的 IAM 角色。有关标签添加策略的更多信息，请参阅《用户指南》中的 为 Amazon 资源添加标签。

除了通过标签标识、组织和跟踪 IAM 资源之外，您还可以在 IAM policy 中使用标签，帮助控制哪些人可以查看并与您的资源交互。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

主题

- 选择 Amazon 标签命名约定
- 在 IAM 和 Amazon STS 中进行标记的规则
- 标记 IAM 用户
- 标记 IAM 角色
- 标记客户管理型策略
- 标记 OpenID Connect（OIDC）身份提供者
- 标记 IAM SAML 身份提供者
- 为 Amazon EC2 角色标记实例配置文件
- 标记服务器证书
- 标记虚拟 MFA 设备
- 在 Amazon STS 中传递会话标签

选择 Amazon 标签命名约定

当您开始将标签附加到您的 IAM 资源时，请谨慎选择标签命名约定。对您的所有 Amazon 标签应用同一约定。如果您在策略中使用标签来控制对 Amazon 资源的访问，这一点尤为重要。如果您已在 Amazon 中使用标签，请检查您的命名约定并相应地进行调整。

> **Note**
>
> 如果您的账户是 Amazon Organizations 的成员，请参阅《Amazon Organizations 用户指南》中的标签策略，以了解有关在 Amazon Organizations 中使用标签的更多信息。

标签命名的最佳实践

以下是标签的一些最佳实践和命名约定。

确保标签名称的使用一致。例如，标签 `CostCenter` 与 `costcenter` 是不同的，因此一个标签可能会被配置为用于财务分析和报告的成本分配标签，而另一个标签可能不会这样配置。同样，`Name` 标签将出现在 Amazon 控制台中，用于很多资源，但 `name` 标签不会。有关标签键和标签键值区分大小写的详细信息，请参阅 Case sensitivity。

许多标签由 Amazon 预定义，或由各种 Amazon 服务自动创建。很多 Amazon 定义的标签名称全部使用小写字母，名称中的单词之间用连字符分隔，前缀用于标识标签的源服务。例如：

- `aws:ec2spot:fleet-request-id` 标识启动实例的 Amazon EC2 竞价型实例请求。
- `aws:cloudformation:stack-name` 标识创建资源的 Amazon CloudFormation 堆栈。
- `elasticbeanstalk:environment-name` 标识创建资源的应用程序。

考虑使用全部小写字母命名标签，用连字符分隔单词，并使用前缀标识组织名称或缩写名称。例如，可以虚构一家名为 AnyCompany 的公司，可以定义如下标签：

- `anycompany:cost-center` 标识内部成本中心代码
- `anycompany:environment-type` 确定环境是开发、测试还是生产环境
- `anycompany:application-id` 标识为其创建资源的应用程序

前缀可以确保将标签明确标识为由贵组织定义，而不是由 Amazon 或您可能正在使用的第三方工具定义。使用所有小写字母和连字符作为分隔符，可以避免对如何大写标签名称产生混淆。例如，`anycompany:project-id` 比 `ANYCOMPANY:ProjectID`、`anycompany:projectID` 或 `Anycompany:ProjectId` 更易记。

在 IAM 和 Amazon STS 中进行标记的规则

大量约定管理 IAM 和 Amazon STS 中标签的创建和应用。

命名标签

为 IAM 资源、Amazon STS 代入角色会话和 Amazon STS 联合身份用户会话制定标签命名约定时，请遵守以下约定：

字符要求 - 标签键和值可以包含字母、数字、空格以及 _ . : / = + - @ 符号的任意组合。

区分大小写 - 标签键是否区分大小写根据标记的 IAM 资源的类型而不同。IAM 用户和角色的标签键值不区分大小写，但会保留大小写。这意味着您不能拥有单独的 **Department** 和 **department** 标签键。如果您已使用 **Department=finance** 标签标记用户并添加 **department=hr** 标签，则它会替换第一个标签。不会添加第二个标签。

对于其他 IAM 资源类型，标签键值区分大小写。这意味着您可以有单独的 **Costcenter** 和 **costcenter** 标签键。例如，如果您使用 **Costcenter = 1234** 标签为客户托管的策略添加了标签，并且添加了 **costcenter = 5678** 标签，策略将同时拥有 **Costcenter** 和 **costcenter** 标签键。

作为最佳实践，我们建议您避免使用大小写处理不一致的类似标签。我们建议您确定利用标签的策略，并在所有资源类型中一致地实施该策略。要了解有关标记最佳实践的更多信息，请参阅《Amazon Web Services 一般参考》中的 标记 Amazon 资源。

以下列表显示了附加到 IAM 资源的标签键在区分大小写方面的差异。

标签键值不区分大小写：

- IAM 角色
- IAM 用户

标签键值区分大小写。

- 客户托管策略
- 实例配置文件
- OpenID Connect 身份提供程序
- SAML 身份提供程序
- 服务器证书
- 虚拟 MFA 设备

此外，以下规则适用：

- 您不能创建以文本 **aws:** 开头的标签键或值。此标签前缀是专为 Amazon 内部使用预留的。
- 您可以使用空值创建标签（如 **phoneNumber =** ）。不能创建空标签键。
- 您不能在单个标签中指定多个值，但可以在单个值中创建自定义多值结构。例如，假设用户 Zhang 在工程团队和 QA 团队工作。如果附加 **team = Engineering** 标签，然后附加 **team = QA** 标

签，则会将标签的值将从 **Engineering** 更改为 **QA**。相反，您可以使用自定义分隔符在单个标签中包含多个值。在此示例中，您可以将 **team = Engineering:QA** 标签附加到 Zhang。

> **Note**
>
> 在此示例中，要使用 **team** 标签控制对工程师的访问，您必须创建一个允许每个可能包含 **Engineering**（包括 **Engineering:QA**）的配置的策略。要了解有关在策略中使用标签的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

应用和编辑标签

在将标签附加到 IAM 资源时遵循以下约定：

- 您可以标记大多数 IAM 资源，但不能标记组、代入的角色、访问报告或基于硬件的 MFA 设备。
- 不能使用标签编辑器标记 IAM 资源。标签编辑器不支持 IAM 标签。有关将标签编辑器与其他服务结合使用的信息，请参阅《Amazon Resource Groups 用户指南》中的 使用标签编辑器。
- 要标记 IAM 资源，您必须拥有特定权限。要标记或取消标记资源，您还必须有权列出标签。有关详细信息，请参阅本页末尾的每个 IAM 资源的主题列表。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。
- 您可以将同一标签应用于多个 IAM 资源。例如，假设您有一个名为 AWS_Development 的部门，有 12 位成员。您可以让 12 位用户和一个角色具有标签键 **department** 以及值 **awsDevelopment**（**department = awsDevelopment**）。您还可以在其他支持标记的服务中的资源上使用相同标签。
- IAM 实体（用户或角色）不能具有同一标签键的多个实例。例如，如果您有一个标签键值对为 **costCenter = 1234** 的用户，则可以附加标签键值对 **costCenter = 5678**。IAM 将 **costCenter** 标签的值更新为 **5678**。
- 要编辑附加到 IAM 实体（用户或角色）的标签，请附加具有新值的标签以覆盖现有标签。例如，假设您有一个标签键值对为 **department = Engineering** 的用户。如果需要将该用户移动到 QA 部门，则可将 **department = QA** 标签键值对附加到该用户。这将导致 **department** 标签键的 **Engineering** 值被替换为 **QA** 值。

标记 IAM 用户

您可以使用 IAM 标签键值对向 IAM 用户添加自定义属性。例如，要向用户添加位置信息，您可以添加标签键 **location** 和标签值 **us_wa_seattle**。或者，也可以使用三种单独的位置标签键值对：**loc-**

country = us、**loc-state = wa** 和 **loc-city = seattle**。您可以使用标签控制用户对资源的访问权限或控制可附加到用户的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记 IAM 用户所需的权限

您必须配置权限以允许 IAM 用户标记其他用户。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

要允许 IAM 用户为特定用户添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 用户的权限策略。使用您的账号并将 *<username>* 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要允许 IAM 用户自行管理标签

将以下语句添加到用户的权限策略以允许用户管理其自己的标签。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
```

```
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::user/${aws:username}"
}
```

要允许 IAM 用户将标签添加到特定用户

将以下语句添加到需要为特定用户添加而不是删除标签的 IAM 用户的权限策略。

> **Note**
>
> `iam:TagUser` 操作要求您也包含 `iam:ListUserTags` 操作。

要使用此策略，请将 *<username>* 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理 IAM 用户的标签（控制台）

您可以从 Amazon Web Services 管理控制台 中管理 IAM 用户的标签。

要管理用户的标签（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Users（用户），然后选择要编辑的用户的名称。

3. 选择 Tags (标签) 选项卡，然后完成下列操作之一：
   - 如果用户还没有标签，请选择添加新标签。
   - 选择 Manage tags（管理标签）来管理一组现有的标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes （保存更改)。

管理 IAM 用户（Amazon CLI 或 Amazon API）的标签

您可以为 IAM 用户列出、附加或删除标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 用户的标签。

要列出当前附加到 IAM 用户的标签（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam list-user-tags
- Amazon API：ListUserTags

将标签附加到 IAM 用户（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-user
- Amazon API：TagUser

从 IAM 用户中删除标签（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam untag-user
- Amazon API：UntagUser

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记 IAM 角色

您可以使用 IAM 标签键值对向 IAM 角色添加自定义属性。例如，要向角色添加位置信息，您可以添加标签键 **`location`** 和标签值 **`us_wa_seattle`**。或者，也可以使用三种单独的位置标签键值对：**`loc-country = us`**、**`loc-state = wa`** 和 **`loc-city = seattle`**。您可以使用标签控制角色对资源的访问权限或控制可附加到角色的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记 IAM 角色所需的权限

您必须配置权限以允许 IAM 角色标记其他实体（用户或角色）。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListRoleTags`
- `iam:TagRole`
- `iam:UntagRole`
- `iam:ListUserTags`
- `iam:TagUser`
- `iam:UntagUser`

要允许 IAM 角色为特定用户添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 角色的权限策略。使用您的账号并将 *<username>* 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser",
        "iam:UntagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要允许 IAM 角色将标签添加到特定用户

将以下语句添加到需要为特定用户添加而不是删除标签的 IAM 角色的权限策略。

要使用此策略，请将 *<username>* 替换为需要管理其标签的用户的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
```

```
    "Effect": "Allow",
    "Action": [
        "iam:ListUserTags",
        "iam:TagUser"
    ],
    "Resource": "arn:aws:iam::<account-number>:user/<username>"
}
```

要运行 IAM 角色为特定角色添加、列出或删除标签

将以下语句添加到需要管理标签的 IAM 角色的权限策略。将 *<rolename>* 替换为需要管理其标签的角色的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListRoleTags",
        "iam:TagRole",
        "iam:UntagRole"
    ],
    "Resource": "arn:aws:iam::<account-number>:role/<rolename>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理 IAM 角色的标签（控制台）

您可以从 Amazon Web Services 管理控制台 中管理 IAM 角色的标签。

要管理角色的标签（控制台）

1. 登录Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择角色，然后选择要编辑的角色的名称。
3. 选择 Tags (标签) 选项卡，然后完成下列操作之一：
   - 如果用户还没有标签，请选择 Add new tag（添加新标签）。
   - 选择 Manage tags（管理标签）来管理一组现有的标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes（保存更改）。

管理 IAM 角色（Amazon CLI 或 Amazon API）的标签

您可以为 IAM 角色列出、附加或删除标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 角色的标签。

列出当前附加到 IAM 角色的标签（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam list-role-tags
- Amazon API：ListRoleTags

将标签附加到 IAM 角色（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-role
- Amazon API：TagRole

从 IAM 角色中删除标签（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam untag-role
- Amazon API：UntagRole

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记客户管理型策略

您可以使用 IAM 标签键值对将自定义属性添加到客户托管策略。例如，要使用部门信息标记策略，您可以添加标签键 **`Department`** 和标签值 **`eng`**。或者，您可能希望标签策略表明它们适用于特定环境，例如 **`Environment = lab`**。您可以使用标签控制对资源的访问权限或控制可附加到资源的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记客户托管策略所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记客户托管策略。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- iam:ListPolicyTags
- iam:TagPolicy
- iam:UntagPolicy

要允许 IAM 实体（用户或角色）添加、列出或删除客户托管策略的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 *<policyname>* 替换为需要管理其标签的策略的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListPolicyTags",
        "iam:TagPolicy",
        "iam:UntagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

要允许 IAM 实体（用户或角色）向特定客户托管策略添加标签

将以下语句添加到需要为特定策略添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> iam:TagPolicy 操作要求您也包含 iam:ListPolicyTags 操作。

要使用此策略，请将 *<policyname>* 替换为需要管理其标签的策略的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListPolicyTags",
        "iam:TagPolicy"
    ],
    "Resource": "arn:aws:iam::<account-number>:policy/<policyname>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理 IAM 客户托管策略的标签（控制台）

您可以从 Amazon Web Services 管理控制台 中管理 IAM 客户托管策略的标签。

要管理客户托管策略的标签（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Policies（策略），然后选择要编辑的客户托管策略的名称。
3. 选择标签选项卡，然后选择管理标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes （保存更改)。

管理 IAM 客户托管策略（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除 IAM 客户托管策略的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM 客户托管策略的标签。

要列出当前附加到 IAM 客户托管策略（Amazon CLI 或 Amazon API）的标签

- Amazon CLI：aws iam list-policy-tags
- Amazon API：ListPolicyTags

要将标签附加到 IAM 客户托管策略（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-policy
- Amazon API：TagPolicy

要从 IAM 客户托管策略（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-policy
- Amazon API：UntagPolicy

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记 OpenID Connect（OIDC）身份提供者

您可以使用 IAM 标签键值向 IAM OpenID Connect (OIDC) 身份提供程序添加自定义属性。例如，要识别 OIDC 身份提供程序，您可以添加标签键 **google** 和标签值 **oidc**。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

标记 IAM OIDC 身份提供程序所需的权限

您必须配置权限以允许 IAM OIDC 实体（用户或角色）标记 IAM OIDC 身份提供程序。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListOpenIDConnectProviderTags`
- `iam:TagOpenIDConnectProvider`
- `iam:UntagOpenIDConnectProvider`

允许 IAM 实体添加、列出或删除 IAM OIDC 身份提供者的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 *<OIDCProviderName>* 替换为需要管理其标签的 OIDC 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider",
        "iam:UntagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

允许 IAM 实体（用户或角色）向特定 IAM OIDC 身份提供程序添加标签

将以下语句添加到需要为特定身份提供程序添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> `iam:TagOpenIDConnectProvider` 操作要求您也包含 `iam:ListOpenIDConnectProviderTags` 操作。

要使用此策略，请将 *<OIDCProviderName>* 替换为需要管理其标签的 OIDC 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListOpenIDConnectProviderTags",
        "iam:TagOpenIDConnectProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:oidc-provider/<OIDCProviderName>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理 IAM OIDC 身份提供程序的标签（控制台）

您可以从 Amazon Web Services 管理控制台 中管理 IAM OIDC 身份提供程序的标签。

要管理 OIDC 身份提供程序的标签（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Identity providers（身份提供程序），然后选择要编辑的身份提供程序的名称。
3. 选择标签选项卡，然后在标签部分中，选择管理标签并完整以下操作之一：
   - 如果 OIDC 身份提供程序还没有标签，请选择 Add tag（添加标签）或添加新标签。
   - 编辑现有的标签键和值。
   - 选择 Remove tag（删除标签）可删除标签。
4. 然后选择 Save changes （保存更改)。

管理 IAM OIDC 身份提供程序（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除 IAM OIDC 身份提供程序的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM OIDC 身份提供程序的标签。

要列出当前附加到 IAM OIDC 身份提供程序（Amazon CLI 或 Amazon API）的标签

- Amazon CLI：aws iam list-open-id-connect-provider-tags
- Amazon API：ListOpenIDConnectProviderTags

要将标签附加到 IAM OIDC 身份提供程序（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-open-id-connect-provider
- Amazon API：TagOpenIDConnectProvider

要从 IAM OIDC 身份提供程序（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-open-id-connect-provider
- Amazon API：UntagOpenIDConnectProvider

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记 IAM SAML 身份提供者

您可以使用 IAM 标签键值对将自定义属性添加到 SAML 身份提供程序。例如，要识别提供商，您可以添加标签键 **`okta`** 和标签值 **`saml`**。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

标记 SAML 身份提供程序所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记基于 SAML 2.0 的身份提供程序 (IdP)。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListSAMLProviderTags`
- `iam:TagSAMLProvider`

- `iam:UntagSAMLProvider`

要允许 IAM 实体（用户或角色）添加、列出或删除 SAML 身份提供程序的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的帐号并将 *<SAMLProviderName>* 替换为需要管理其标签的 SAML 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider",
        "iam:UntagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

要允许 IAM 实体（用户或角色）向特定的 SAML 身份提供程序添加标签

将以下语句添加到需要为特定 SAML 提供商添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> `iam:TagSAMLProvider` 操作要求您也包含 `iam:ListSAMLProviderTags` 操作。

要使用此策略，请将 *<SAMLProviderName>* 替换为需要管理其标签的 SAML 提供商的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListSAMLProviderTags",
        "iam:TagSAMLProvider"
    ],
    "Resource": "arn:aws:iam::<account-number>:saml-provider/<SAMLProviderName>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理 IAM SAML 身份提供程序的标签（控制台）

您可以从 Amazon Web Services 管理控制台 中管理 IAM SAML 身份提供程序的标签。

要管理 SAML 身份提供程序的标签（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在控制台的导航窗格中，选择 Identity providers（身份提供程序），然后选择要编辑的 SAML 身份提供程序的名称。
3. 选择标签选项卡，然后在标签部分中，选择管理标签并完整以下操作之一：
   - 如果 SAML 身份提供程序还没有标签，请选择 Add tag（添加标签），或者添加新标签。
   - 编辑现有的标签键和值。
   - 选择 Remove tag（删除标签）可删除标签。
4. 添加或删除标签以完成标签集。然后选择 Save changes （保存更改)。

管理 IAM SAML 身份提供程序（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除 IAM SAML 身份提供程序的标签。您可以使用 Amazon CLI 或 Amazon API 管理 IAM SAML 身份提供程序的标签。

要列出当前附加到 SAML 身份提供程序（Amazon CLI 或 Amazon API）的标签

- Amazon CLI：aws iam list-saml-provider-tags
- Amazon API：ListSAMLProviderTags

要将标签附加到 SAML 身份提供程序（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-saml-provider
- Amazon API：TagSAMLProvider

要从 SAML 身份提供程序（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-saml-provider
- Amazon API：UntagSAMLProvider

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

为 Amazon EC2 角色标记实例配置文件

当您启动 Amazon EC2 实例时，可指定要与实例关联的 IAM 角色。实例配置文件是 IAM 角色的容器，可用来在实例启动时将角色信息传递给 Amazon EC2 实例。您可以在使用 Amazon CLI 或 Amazon API 时标记实例配置文件。

您可以使用 IAM 标签键值对将自定义属性添加到实例配置文件。例如，要将部门信息添加到实例配置文件，您可以添加标签键 **access-team** 和标签值 **eng**。这样做可以让具有匹配标签的主体访问具有相同标签的实例配置文件。您可以使用多个标签键值对来指定团队和项目：**access-team = eng** 和 **project = peg**。您可以使用标签控制用户对资源的访问权限或控制可附加到用户的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记实例配置文件所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记实例配置文件。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListInstanceProfileTags`
- `iam:TagInstanceProfile`
- `iam:UntagInstanceProfile`

要允许 IAM 实体（用户或角色）添加、列出或删除实例配置文件的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 *<InstanceProfileName>* 替换为需要管理其标签的实例配置文件的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
```

```
        "iam:ListInstanceProfileTags",
        "iam:TagInstanceProfile",
        "iam:UntagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

允许 IAM 实体（用户或角色）向特定实例配置文件添加标签

将以下语句添加到需要为特定实例配置文件添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> iam:TagInstanceProfile 操作要求您也包含 iam:ListInstanceProfileTags 操作。

要使用此策略，请将 *<InstanceProfileName>* 替换为需要管理其标签的实例配置文件的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListInstanceProfileTags",
        "iam:TagInstanceProfile"
    ],
    "Resource": "arn:aws:iam::<account-number>:instance-profile/<InstanceProfileName>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理实例配置文件（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除实例配置文件的标签。您可以使用 Amazon CLI 或 Amazon API 管理实例配置文件的标签。

要列出当前附加到实例配置文件的标签（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam list-instance-profile-tags
- Amazon API：ListInstanceProfileTags

要将标签附加到实例配置文件（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-instance-profile
- Amazon API：TagInstanceProfile

要从实例配置文件（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-instance-profile
- Amazon API：UntagInstanceProfile

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记服务器证书

如果您使用 IAM 管理 SSL/TLS 证书，则可以使用 Amazon CLI 或 Amazon API 标记 IAM 中的服务器证书。对于 Amazon Certificate Manager (ACM) 支持的区域中的证书，我们建议您使用 ACM 而不是 IAM 预置、管理和部署您的服务器证书。在不支持的区域中，您必须将 IAM 作为证书管理器。要了解 ACM 支持的具体区域，请参阅《Amazon Web Services 一般参考》中的 Amazon Certificate Manager 端点和限额。

您可以使用 IAM 标签键值对向服务器证书添加自定义属性。例如，要添加有关服务器证书拥有者者或管理员的信息，请添加标签键 **owner** 和标签值 **net-eng**。或者，您可以通过添加标签键 **CostCenter** 和标签值 **1234** 来指定成本中心。您可以使用标签控制对资源的访问权限或控制可附加到资源的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记服务器证书所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记服务器证书。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- `iam:ListServerCertificateTags`
- `iam:TagServerCertificate`

- iam:UntagServerCertificate

要允许 IAM 实体（用户或角色）添加、列出或删除服务器证书的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账户并将 *<CertificateName>* 替换为需要管理其标签的服务器证书的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate",
        "iam:UntagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

要允许 IAM 实体（用户或角色）向特定服务器证书添加标签

将以下语句添加到需要为特定服务器证书添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> iam:TagServerCertificate 操作要求您也包含 iam:ListServerCertificateTags 操作。

要使用此策略，请将 *<CertificateName>* 替换为需要管理其标签的服务器证书的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListServerCertificateTags",
        "iam:TagServerCertificate"
    ],
    "Resource": "arn:aws:iam::<account-number>:server-certificate/<CertificateName>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理服务器证书（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除服务器证书的标签。您可以使用 Amazon CLI 或 Amazon API 管理服务器证书的标签。

要列出当前附加到服务器证书（Amazon CLI 或 Amazon API）的标签

- Amazon CLI：aws iam list-server-certificate-tags
- Amazon API：ListServerCertificateTags

要将标签附加到服务器证书（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-server-certificate
- Amazon API：TagServerCertificate

要从服务器证书（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-server-certificate
- Amazon API：UntagServerCertificate

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

标记虚拟 MFA 设备

您可以使用 IAM 标签键值对向虚拟 MFA 设备添加自定义属性。例如，要为用户的虚拟 MFA 设备添加成本中心信息，可以添加标签键 **`CostCenter`** 和标签值 **`1234`**。您可以使用标签控制对资源的访问权限或控制可附加到对象的标签。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您还可以在 Amazon STS 中使用标签，以在代入角色或联合身份用户身份时添加自定义属性。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。

标记虚拟 MFA 设备所需的权限

您必须配置权限以允许 IAM 实体（用户或角色）标记虚拟 MFA 设备。您可以在 IAM policy 中指定以下一项或所有 IAM 标签操作：

- iam:ListMFADeviceTags
- iam:TagMFADevice
- iam:UntagMFADevice

要允许 IAM 实体（用户或角色）添加、列出或删除虚拟 MFA 设备的标签

将以下语句添加到需要管理标签的 IAM 实体的权限策略。使用您的账号并将 *<MFATokenID>* 替换为需要管理其标签的虚拟 MFA 设备的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice",
        "iam:UntagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

要允许 IAM 实体（用户或角色）向特定虚拟 MFA 设备添加标签

将以下语句添加到需要为特定 MFA 设备添加而不是删除标签的 IAM 实体的权限策略。

> **Note**
>
> iam:TagMFADevice 操作要求您也包含 iam:ListMFADeviceTags 操作。

要使用此策略，请将 *<MFATokenID>* 替换为需要管理其标签的虚拟 MFA 设备的名称。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

```
{
    "Effect": "Allow",
    "Action": [
        "iam:ListMFADeviceTags",
        "iam:TagMFADevice"
    ],
    "Resource": "arn:aws:iam::<account-number>:mfa/<MFATokenID>"
}
```

或者，也可以使用 Amazon 托管策略（如 IAMFullAccess）来提供对 IAM 的完全访问权限。

管理虚拟 MFA 设备（Amazon CLI 或 Amazon API）的标签

您可以列出、附加或删除虚拟 MFA 设备的标签。您可以使用 Amazon CLI 或 Amazon API 管理虚拟 MFA 设备的标签。

要列出当前附加到虚拟 MFA 设备（Amazon CLI 或 Amazon API）的标签

- Amazon CLI：aws iam list-mfa-device-tags
- Amazon API：ListMFADeviceTags

要将标签附加到虚拟 MFA 设备（Amazon CLI 或 Amazon API）

- Amazon CLI：aws iam tag-mfa-device
- Amazon API：TagMFADevice

要从虚拟 MFA 设备（Amazon CLI 或 Amazon API）中删除标签

- Amazon CLI：aws iam untag-mfa-device
- Amazon API：UntagMFADevice

有关将标签附加到其他 Amazon 服务的资源的信息，请参阅这些服务的文档。

有关使用标签通过 IAM 权限策略设置更精细权限的信息，请参阅 IAM policy 元素：变量和标签。

在 Amazon STS 中传递会话标签

会话标签是您在代入 IAM 角色或联合 Amazon STS 用户身份时传递的键/值对属性。您可以通过 Amazon STS 或您的身份提供者 (IdP) 发出 Amazon CLI 或 Amazon API 请求来完成此操作。当您使用 Amazon STS 请求临时安全凭证时，会生成会话。会话过期并具有凭证，例如访问密钥对和会话令牌。当您使用会话凭证发出后续请求时，请求上下文包含 `aws:PrincipalTag` 上下文密钥。您可以使用策略的 `Condition` 元素中的 `aws:PrincipalTag` 键，基于这些标签来允许或拒绝访问。

当您使用临时凭证发出请求时，您的主体可能会包含一组标签。这些标签来自以下来源：

1. 会话标签 - 这些标签在您使用 Amazon CLI 或 Amazon API 代入角色或联合身份用户身份时传递。有关这些操作的更多信息，请参阅 会话标记操作。

2. 传入的可传递会话标签 - 这些标签继承自角色链中的上一个会话。有关更多信息，请参阅本主题后面的使用会话标签链接角色。
3. IAM 标签 — 附加到您的 IAM 所担任角色的标签。

主题

- 会话标记操作
- 关于会话标签的需知信息
- 添加会话标签所需的权限
- 使用 AssumeRole 传递会话标签
- 使用 AssumeRoleWithSAML 传递会话标签
- 使用 AssumeRoleWithWebIdentity 传递会话标签
- 使用 GetFederationToken 传递会话标签
- 使用会话标签链接角色
- 将会话标签用于 ABAC
- 在 CloudTrail 中查看会话标签

会话标记操作

您可以在 Amazon STS 中使用以下 Amazon CLI 或 Amazon API 操作来传递会话标签。Amazon Web Services 管理控制台 切换角色 功能不支持传递会话标签。

您还可以将会话标签设置为可传递。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅 使用会话标签链接角色。

下表比较了传递会话标签的方法。

操作	谁可以担任角色	传递标签的方法	设置可传递标签的方法
`assume-role` CLI 或 `AssumeRole` API 操作	IAM 用户或会话	`Tags` API 参数或 `--tags` CLI 选项	`TransitiveTagKeys` API 参数或 `--transitive-tag-keys` CLI 选项

操作	谁可以担任角色	传递标签的方法	设置可传递标签的方法
assume-role-with-saml CLI 或 AssumeRoleWithSAML API 操作	使用 SAML 身份提供者验证身份的任何用户	PrincipalTag SAML 属性	TransitiveTagKeys SAML 属性
assume-role-with-web-identity CLI 或 AssumeRoleWithWebIdentity API 操作	使用 OIDC 提供者验证身份的任何用户	PrincipalTag OIDC 令牌	TransitiveTagKeys OIDC 令牌
get-federation-token CLI 或 GetFederationToken API 操作	IAM 用户或根用户	Tags API 参数或 --tags CLI 选项	不支持

如果满足以下条件，则支持会话标记的操作会失败：

- 您传递的会话标签数超过 50。
- 会话标签键的纯文本超过 128 个字符。
- 会话标签值的纯文本超过 256 个字符。
- 会话策略纯文本的总大小超过 2048 个字符。
- 会话策略和会话标签的组合包总大小太大。如果操作失败，则错误消息以百分比形式指示策略和标签的组合接近大小上限的程度。

关于会话标签的需知信息

在您使用会话标签之前，请查看以下有关会话和标签的详细信息。

- 使用会话标签时，连接到传递标签的身份提供者 (IdP) 的所有角色的信任策略都必须具有 `sts:TagSession` 权限。对于信任策略中没有此权限的角色，`AssumeRole` 操作会失败。
- 请求会话时，可以将主体标记指定为会话标记。标签适用于您使用会话凭证发出的请求。
- 会话标签使用键/值对。例如，要向会话添加联系信息，您可以添加会话标签键 `email` 和标签值 `johndoe@example.com`。
- 会话标签必须遵循在 IAM 和 Amazon STS 中命名标签的规则。本主题包括有关适用于会话标签的区分大小写和受限制前缀的信息。
- 新的会话标签将使用相同的标签键覆盖现有的代入角色或联合用户会话标签，无论字符大小写如何。
- 您无法使用 Amazon Web Services 管理控制台传递会话标签。
- 会话标签仅对当前会话有效。
- 会话标签支持角色链。默认情况下，Amazon STS 不传递到后续角色会话。不过，您可以将会话标签设置为可传递。传递标签在角色链期间持续存在，在角色信任策略评估之后，替换匹配 `ResourceTag` 的值。有关更多信息，请参阅 使用会话标签链接角色。
- 您可以使用会话标签来控制对资源的访问，或者控制哪些标签可以传递到后续会话。有关更多信息，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。
- 您可以在 Amazon CloudTrail 日志中查看会话的主体标签，包括其会话标签。有关更多信息，请参阅 在 CloudTrail 中查看会话标签。
- 您必须为每个会话标签传递一个值。Amazon STS 不支持多值会话标签。
- 您最多可以传递 50 个会话标签。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。
- Amazon 转换会将传递的会话策略和会话标签合并压缩为具有单独限制的打包二进制格式。如果超过此限制，Amazon CLI 或 Amazon API 错误消息以百分比形式指示策略和标签的组合接近大小上限的程度。

添加会话标签所需的权限

除了与 API 操作匹配的操作之外，您的策略中还必须具有以下仅限授权执行的操作：

```
sts:TagSession
```

> **Important**
>
> 使用会话标签时，连接到身份提供者 (IdP) 的所有角色的角色信任策略都必须具有 `sts:TagSession` 权限。对于连接到 IdP 的任何角色，如果在没有此权限的情况下传递会话

标签，AssumeRole 操作将失败。如果您不想更新每个角色的角色信任策略，则可以使用单独的 IdP 实例传递会话标签。然后仅将 sts:TagSession 权限添加到连接至单独 IdP 的角色。

您可以将 sts:TagSession 操作与以下条件键搭配使用。

- aws:PrincipalTag – 使用此键可将附加到发出请求的主体的标签与您在策略中指定的标签进行比较。例如，您可以仅在发出请求的主体具有指定的标签时，才允许主体传递会话标签。
- aws:RequestTag – 使用此键可将请求中传递的标签键/值对与您在策略中指定的标签对进行比较。例如，您可以允许主体传递指定的会话标签，但只能使用指定的值。
- aws:ResourceTag - 使用此键可将您在策略中指定的标签键/值对与附加到资源的键/值对进行比较。例如，您可以仅在当主体代入的角色包含指定标签时，才允许主体传递会话标签。
- aws:TagKeys – 将请求中的标签键与您在策略中指定的键进行比较。例如，您可以只允许主体传递具有指定标签键的会话标签。此条件键限制可传递的最大一组会话标签。
- sts:TransitiveTagKeys - 将请求中的可传递会话标签键与在策略中指定的键进行比较。例如，您可以编写一个策略，仅允许主体将特定标签设置为可传递标签。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅 使用会话标签链接角色。

例如，以下角色信任策略允许 test-session-tags 用户代入策略附加到的角色。用户在代入该角色时，必须使用 Amazon CLI 或 Amazon API 传递三个所需的会话标签和所需的外部 ID。此外，用户可以选择将 Project 和 Department 标签设置为可传递。

Example会话标签的角色信任策略示例

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIamUserAssumeRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-
tags"},
            "Condition": {
              "StringLike": {
                    "aws:RequestTag/Project": "*",
```

```
                    "aws:RequestTag/CostCenter": "*",
                    "aws:RequestTag/Department": "*"
                },
                "StringEquals": {"sts:ExternalId": "Example987"}
            }
        },
        {
            "Sid": "AllowPassSessionTagsAndTransitive",
            "Effect": "Allow",
            "Action": "sts:TagSession",
            "Principal": {"AWS": "arn:aws:iam::123456789012:user/test-session-
tags"},
            "Condition": {
                "StringLike": {
                    "aws:RequestTag/Project": "*",
                    "aws:RequestTag/CostCenter": "*"
                },
                "StringEquals": {
                    "aws:RequestTag/Department": [
                        "Engineering",
                        "Marketing"
                    ]
                },
                "ForAllValues:StringEquals": {
                    "sts:TransitiveTagKeys": [
                        "Project",
                        "Department"
                    ]
                }
            }
        }
    ]
}
```

此策略有何作用？

- AllowIamUserAssumeRole 语句允许 test-session-tags 用户代入策略所附加到的角色。用户在代入该角色时，必须传递所需的会话标签和外部 ID。
  - 此语句的第一个条件块要求用户传递 Project、CostCenter 和 Department 会话标签。标签值在此语句中无关紧要，因此我们为标签值使用通配符 (*)。此块确保用户至少传递这三个会话标签。否则，该操作将失败。用户可以传递其他标签。

  - 第二个条件块要求用户传递值为 Example987 的外部 ID。
- 该 AllowPassSessionTagsAndTransitive 语句允许 sts:TagSession 仅限授权执行的操作。必须允许此操作，然后用户才能传递会话标签。如果您的策略包含第一个语句而没有第二个语句，则用户无法代入角色。
  - 此语句的第一个条件块允许用户为 CostCenter 和 Project 会话标签传递任何值。您可以通过在策略中为标签值使用通配符 (*) 来执行此操作，这需要使用 StringLike 条件运算符。
  - 第二个条件块仅允许用户为 Department 会话标签传递 Engineering 或 Marketing 值。
  - 第三个条件块列出可以设置为可传递的最大一组标签。用户可以选择将一部分标签设置为可传递，也可以不将任何标签设置为可传递。他们不能将其他标签设置为可传递。您可以通过添加包含 "Null":{"sts:TransitiveTagKeys":"false"} 的其他条件块，将至少一个标签设置为可传递。

使用 AssumeRole 传递会话标签

AssumeRole 操作返回一组可用于访问 Amazon 资源的临时凭证。您可以使用 IAM 用户或角色凭证来调用 AssumeRole。要在代入角色时传递会话标签，请使用 --tags Amazon CLI 选项或 Tags Amazon API 参数。

要将标签设置为可传递，请使用 --transitive-tag-keys Amazon CLI 选项或 TransitiveTagKeys Amazon API 参数。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅 使用会话标签链接角色。

以下示例显示使用 AssumeRole 的示例请求。在此示例中，在代入 my-role-example 角色时，您将创建一个名为 my-session 的会话。添加会话标签键/值对 Project = Automation、CostCenter = 12345 和 Department = Engineering。您还可以通过指定 Project 和 Department 标签的键，将其设置为可传递标签。您必须为每个会话标签传递一个值。Amazon STS 不支持多值会话标签。

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/my-role-example \
--role-session-name my-session \
--tags Key=Project,Value=Automation Key=CostCenter,Value=12345
 Key=Department,Value=Engineering \
--transitive-tag-keys Project Department \
--external-id Example987
```

使用 AssumeRoleWithSAML 传递会话标签

AssumeRoleWithSAML 操作使用基于 SAML 的联合身份验证进行身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 SAML 的联合身份验证用于 Amazon Web Services 管理控制台访问的更多信息，请参阅使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台。有关 Amazon CLI 或 Amazon API 访问的详细信息，请参阅SAML 2.0 联合身份验证。有关为 Active Directory 用户设置 SAML 联合身份的教程，请参阅 Amazon 安全博客 中的使用 Active Directory 联合身份验证服务 (ADFS) 的 Amazon 联合身份验证。

作为管理员，您可以使用 Amazon STS AssumeRoleWithSAML 操作，允许公司目录的成员联合身份到 Amazon 中。要执行此操作，您必须完成以下任务：

1. 将网络配置为适用于 Amazon 的 SAML 提供商。
2. 在 IAM 中创建 SAML 提供商
3. 创建用于 SAML 2.0 联合身份验证的角色（控制台）
4. 完成配置 SAML IdP 并为 SAML 身份验证响应创建断言

Amazon 包括身份提供者，通过其身份解决方案提供经认证的端到端会话标签体验。要了解如何使用这些身份提供者配置会话标签，请参阅将第三方 SAML 解决方案提供者与 Amazon 集成。

要将 SAML 属性作为会话标签传递，请包含 Attribute 元素并将 Name 属性设置为 https://aws.amazon.com/SAML/Attributes/PrincipalTag:*{TagKey}*。使用 AttributeValue 元素指定标签的值。为每个会话标签包含一个单独的 Attribute 元素。

例如，假设您要将以下身份属性作为会话标记传递：

- Project:Automation
- CostCenter:12345
- Department:Engineering

要传递这些属性，请在 SAML 断言中包含以下元素。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Project">
  <AttributeValue>Automation</AttributeValue>
</Attribute>
```

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:CostCenter">
  <AttributeValue>12345</AttributeValue>
</Attribute>
<Attribute Name="https://aws.amazon.com/SAML/Attributes/PrincipalTag:Department">
  <AttributeValue>Engineering</AttributeValue>
</Attribute>
```

要将上述标签设置为可传递，请包含另一个 `Attribute` 元素并将 `Name` 属性设置为 `https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys`。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅 使用会话标签链接角色。

要将 `Project` 和 `Department` 标签设置为可传递，请使用以下多值属性。

Example SAML 断言的示例片段

```
<Attribute Name="https://aws.amazon.com/SAML/Attributes/TransitiveTagKeys">
  <AttributeValue>Project</AttributeValue>
  <AttributeValue>Department</AttributeValue>
</Attribute>
```

使用 AssumeRoleWithWebIdentity 传递会话标签

使用 OpenID Connect（OIDC）兼容的联合身份验证进行 `AssumeRoleWithWebIdentity` 操作的身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。有关将基于 Web 身份联合验证用于 Amazon Web Services 管理控制台访问的更多信息，请参阅OIDC 联合身份验证。

要从 OpenID Connect（OIDC）传递会话标签，您必须在提交 `AssumeRoleWithWebIdentity` 请求时在 JSON Web 令牌（JWT）中包含会话标签。要了解有关 OIDC 令牌和声明的更多信息，请参阅《Amazon Cognito 开发人员指南》中的将令牌与用户群体结合使用。

对于在 JWT 中包含会话标签的情况，Amazon 支持两种声明格式：

- 嵌套声明格式
- 展平声明格式

嵌套声明格式

嵌套声明格式使用 JWT `https://aws.amazon.com/tags` 命名空间内的结构。在这种格式中：

- 主体标签作为 `principal_tags` 键下的嵌套对象显示。

- 每个主体标签都是单独的字符串值。
- 可传递标签键在 transitive_tag_keys 键下方的数组中显示。
- principal_tags 和 transitive_tag_keys 都嵌套在 https://aws.amazon.com/tags 命名空间下。

以下示例演示了使用嵌套对象格式的解码 JWT：

Example使用嵌套声明格式的解码 JSON Web 令牌示例

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags": {
        "principal_tags": {
            "Project": ["Automation"],
            "CostCenter": ["987654"],
            "Department": ["Engineering"]
        },
        "transitive_tag_keys": [
            "Project",
            "CostCenter"
        ]
    }
}
```

展平声明格式

展平声明格式与 Microsoft Entra ID 等不支持 JWT 声明中嵌套对象的身份提供商兼容。在这种格式中：

- 主体标签显示为带有前缀 https://aws.amazon.com/tags/principal_tags/ 的独立声明。
- 每个主体标签都是单独的字符串值。
- 在单个声明中，可传递标签键作为带有前缀 https://aws.amazon.com/tags/transitive_tag_keys 的字符串数组显示。

接下来看看如何使用展平声明格式来表示相同的信息：

Example使用展平声明格式的解码 JSON Web 令牌示例

```
{
    "sub": "johndoe",
    "aud": "ac_oic_client",
    "jti": "ZYUCeRMQVtqHypVPWAN3VB",
    "iss": "https://xyz.com",
    "iat": 1566583294,
    "exp": 1566583354,
    "auth_time": 1566583292,
    "https://aws.amazon.com/tags/principal_tags/Project": "Automation",
    "https://aws.amazon.com/tags/principal_tags/CostCenter": "987654",
    "https://aws.amazon.com/tags/principal_tags/Department": "Engineering",
    "https://aws.amazon.com/tags/transitive_tag_keys": [
        "Project",
        "CostCenter"
    ]
}
```

两个解码 JWT 示例都显示了使用 `Project`、`CostCenter` 和 `Department` 会话标签对 `AssumeRoleWithWebIdentity` 的调用。两个令牌都将 `Project` 和 `CostCenter` 标签设置为可传递状态。在角色链的串联过程中，可传递标签持续存在。有关更多信息，请参阅 使用会话标签链接角色。

展平声明格式可与嵌套声明格式达成相同结果，但前者针对标签使用展平结构。通过展平声明格式，在 JWT 声明中不支持嵌套 JSON 对象的环境中，您可以包含会话标签。使用任一格式时，务必确保身份提供商已配置为可发布具有相应声明结构的令牌。Amazon 支持两种声明格式，因此您可以选择最符合身份提供商具体要求的格式。

使用 GetFederationToken 传递会话标签

`GetFederationToken` 允许您对用户进行联合身份验证。此操作返回一组可用于访问 Amazon 资源的临时凭证。要将标签添加到联合身份用户会话，请使用 `--tags` Amazon CLI 选项或 `Tags` Amazon API 参数。在使用 `GetFederationToken` 时，您不能将会话标签设置为可传递，因为您无法使用临时凭证担任角色。在这种情况下，您不能使用角色链。

下面的示例演示使用 `GetFederationToken` 的示例请求。在此示例中，在请求令牌时，您创建一个名为 `my-fed-user` 的会话。添加会话标签键/值对 `Project` = `Automation` 和 `Department` = `Engineering`。

Example GetFederationToken CLI 请求示例

```
aws sts get-federation-token \
--name my-fed-user \
--tags key=Project,value=Automation key=Department,value=Engineering
```

当您使用 GetFederationToken 操作返回的临时凭证时，会话的主体标签包括用户的标签以及传递的会话标签。

使用会话标签链接角色

您可以代入一个角色，然后使用临时凭证代入另一个角色。您可以从一个会话继续到另一个会话来使用角色。这就是所说的角色链。当您在代入某个角色的同时传递会话标签，您可以将键设置为可传递。这可确保将这些会话标签传递到角色链中的后续会话。您无法将角色标签设置为可传递。要将这些标签传递给后续会话，请将其指定为会话标签。

> **Note**
>
> 传递标签在角色链期间持续存在，在角色信任策略评估之后，替换匹配 ResourceTag 的值。

以下示例帮助您了解 Amazon STS 如何在角色链中，将会话标签、可传递标签和角色标签传递到后续会话。

在以下示例角色链场景中，您在 Amazon CLI 中使用 IAM 用户的访问密钥代入名为 Role1 的角色。然后，您可以使用生成的会话凭证代入名为 Role2 的第二个角色。接下来，您可以使用第二个会话凭证代入名为 Role3 的第三个角色。这些请求作为三个单独的操作发生。每个角色均已在 IAM 中标记。在每个请求过程中，您都会传递额外的会话标签。
角色链。

链接角色时，您可以确保来自较早会话的标签保留到后续会话。要使用 assume-role CLI 命令执行此操作，您必须将标签作为会话标签传递，并将标签设置为可传递。您传递标签 Star = 1 作为会话标签。此命令还将标签 Heart = 1 附加到角色，并在您使用会话时作为主体标签应用。但是，您还希望自动将 Heart = 1 标签传递给第二个或第三个会话。为此，您可以手动将其包含为会话标签。生成的会话主体标记包括这两个标记，并将它们设置为可传递。
代入角色链中的第一个角色。

您可以使用以下 Amazon CLI 命令执行此请求：

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role1 \
--role-session-name Session1 \
--tags Key=Star,Value=1 Key=Heart,Value=1 \
--transitive-tag-keys Star Heart
```

然后，您可以使用该会话的凭证代入 Role2。此命令将标签 Sun = 2 附加到第二个角色，并在您使用第二个会话时作为主体标签应用。Heart 和 Star 标签继承第一个会话中的可传递会话标签。第二个会话产生的主体标签为 Heart = 1、Star = 1 和 Sun = 2。Heart 和 Star 将继续为可传递标签。附加到 Role2 的 Sun 标签不会标记为可传递，因为它不是会话标签。以后的会话不继承此标签。
代入角色链中的第二个角色。

您可以使用以下 Amazon CLI 命令执行第二个请求：

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role2 \
--role-session-name Session2
```

然后，您使用第二个会话凭证来代入 Role3。第三个会话的主体标签来自任意新的会话标签、继承的可传递会话标签和角色标签。第二个会话上的 Heart = 1 和 Star = 1 标签继承自第一个会话中的可传递会话标签。如果您尝试传递 Sun = 2 会话标记，操作会失败。继承的 Star = 1 会话标签覆盖角色的 Star = 3 标签。在角色链中，传递标签的值将在评估角色信任策略之后，覆盖匹配 ResourceTag 值的角色。在此示例中，如果 Role3 在角色信任策略中将 Star 用作 ResourceTag，并将 ResourceTag 值设置为调用角色会话的可传递标签值。角色的 Lightning 标签也应用到第三个会话，且未设置为可传递。
代入角色链中的第三个角色。

您使用以下 Amazon CLI 命令执行第三个请求：

Example示例 AssumeRole CLI 请求

```
aws sts assume-role \
--role-arn arn:aws:iam::123456789012:role/Role3 \
--role-session-name Session3
```

将会话标签用于 ABAC

基于属性的访问权限控制 (ABAC) 是一种授权策略，该策略基于标签属性来定义权限。

如果您的公司使用 OIDC 基于 SAML 的身份提供者 (IdP) 管理用户实体，则可以配置 SAML 断言，将会话标记传递给 Amazon。例如，对于公司用户身份，当您的员工联合到 Amazon，Amazon 会将其属性应用于其生成的主体。然后，您可以使用 ABAC 来允许或拒绝基于这些属性的权限。有关更多信息，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。

有关将 IAM Identity Center 与 ABAC 配合使用的更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的访问控制的属性。

在 CloudTrail 中查看会话标签

您可以使用 Amazon CloudTrail 查看为代入角色或联合身份用户而发出的请求。CloudTrail 日志文件包括有关所代入角色或联合用户会话的主体标签的信息。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。

例如，假定您发出 Amazon STS `AssumeRoleWithSAML` 请求，传递会话标签，然后将这些标签设置为可传递标签。您可以在 CloudTrail 日志中找到以下信息。

Example AssumeRoleWithSAML CloudTrail 日志示例

```
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyRoleSessionName",
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn"
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
        "durationSeconds": 3600,
        "roleArn": "arn:aws:iam::123456789012:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::123456789012:saml-provider/Shibboleth"
    },
```

您可以查看以下示例 CloudTrail 日志以查看使用会话标签的事件。

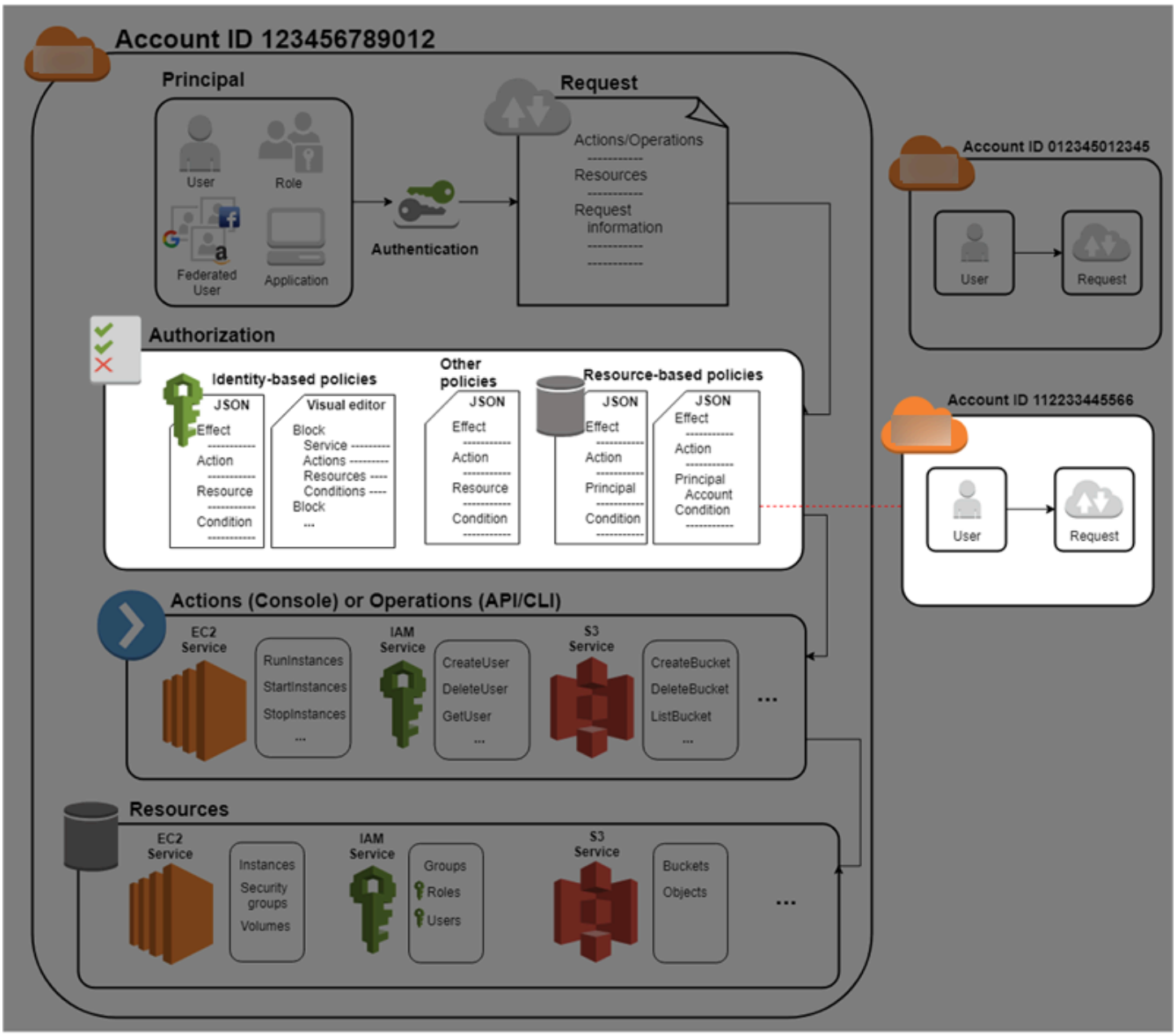
- CloudTrail 日志文件中 Amazon STS 角色链接 API 事件示例

- CloudTrail 日志文件中 SAML Amazon STS API 事件的示例
- CloudTrail 日志文件中 OIDC Amazon STS API 事件示例

适用于 Amazon 资源的 Access Management

Amazon Identity and Access Management (IAM) 是一种 Web 服务，可以帮助您安全地控制对 Amazon 资源的访问。当主体在 Amazon 中发出请求时，Amazon 执行代码会检查是否对主体进行身份验证（登录）和授权（具有权限）。您将创建策略并将其附加到 IAM 身份或 Amazon 资源，以便管理 Amazon 中的访问。策略是 Amazon 中的 JSON 文档，在附加到身份或资源时，策略定义它们的权限。有关策略类型和用法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

有关身份验证和授权过程的其余部分的详细信息，请参阅IAM 的工作原理。



在授权期间，Amazon 执行代码使用请求上下文中的值检查匹配的策略并确定是允许还是拒绝请求。

Amazon 检查应用于请求上下文的每个策略。如果一个策略拒绝请求，Amazon 将拒绝整个请求并停止评估策略。这称为显式拒绝。由于请求是默认拒绝的，因此，只有在适用的策略允许请求的每个部分时，IAM 才会授权请求。单个账户中对于请求的评估逻辑遵循以下规则：

- 默认情况下，所有请求都被隐式拒绝。（或者，默认情况下，Amazon Web Services 账户根用户 拥有完全访问权限。）
- 基于身份或基于资源的策略中的显式允许将覆盖此默认值。

- 如果存在权限边界、Amazon Organizations SCP 或会话策略，它可能会使用隐式拒绝覆盖允许。
- 任何策略中的显式拒绝将覆盖任何允许。

在对您的请求进行身份验证和授权后，Amazon 将批准该请求。如果您需要在另一个账户中发出请求，其他账户中的策略必须允许访问资源。此外，您用于发出请求的 IAM 实体必须具有基于身份的策略，该策略允许该请求。

访问管理资源

有关权限和创建策略的更多信息，请参阅以下资源：

Amazon 安全博客中的以下文章介绍 Amazon S3 存储桶和对象访问策略的常用编写方法。

- 编写 IAM 策略：如何授予对 Amazon S3 存储桶的访问权限
- 编写 IAM 策略：授予对 Amazon S3 存储桶中用户特定文件夹的访问权限
- IAM 策略、存储桶策略和 ACL！天哪！(Controlling Access to S3 Resources)（控制对 S3 资源的访问）。
- RDS 资源级别权限读本
- 阐明 EC2 资源级权限

Amazon Identity and Access Management 中的策略和权限

在 Amazon 中通过创建策略并将其附加到 IAM 身份（用户、用户组或角色）或 Amazon 资源来管理访问权限。策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个 IAM 主体（用户或角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略作为 JSON 文档存储在 Amazon 中。Amazon 支持九种类型的策略：基于身份的策略、基于资源的策略、VPC 端点策略、权限边界、Amazon Organizations 服务控制策略（SCP）、Amazon Organizations 资源控制策略（RCP）、访问控制列表（ACL）、Resource Access Manager (RAM) 共享和会话策略。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，如果一个策略允许 GetUser 操作，则具有该策略的用户可以从 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 获取用户信息。在创建 IAM 用户时，您可以选择允许控制台或编程访问。如果允许控制台访问，则 IAM 用户可以使用其登录凭证登录到控制台。如果允许编程访问，则用户可以通过访问密钥来使用 CLI 或 API。

策略类型

以下策略类型按从最常用到不常用的顺序列出，可在 Amazon 中使用。有关更多详细信息，请参阅下面有关各种策略类型的各部分。

- 基于身份的策略 – 将托管策略和内联策略附加到 IAM 身份（用户、用户所属组或角色）。基于身份的策略向身份授予权限。
- 基于资源的策略 - 将内联策略附加到资源。基于资源的策略的最常见示例是 Amazon S3 存储桶策略和 IAM 角色信任策略。基于资源的策略向在策略中指定的主体授予权限。主体可以与资源位于同一个账户中，也可以位于其他账户中。
- VPC 端点策略 — 附加到 VPC 终端节点，从而控制哪些主体可使用 VPC 端点策略，以及可通过该端点访问哪些资源。VPC 端点策略作为额外的访问边界，其范围仅作用于通过端点的流量。
- 权限边界 - 使用托管策略作为 IAM 实体（用户或角色）的权限边界。该策略定义基于身份的策略可以授予实体的最大权限，但不授予权限。权限边界不定义基于资源的策略可以授予实体的最大权限。
- Amazon Organizations SCP：使用 Amazon Organizations 服务控制策略（SCP）为组织或组织单元（OU）的账户的 IAM 用户和 IAM 角色定义最大权限。SCP 限制基于身份的策略或基于资源的策略授予账户中的 IAM 用户或 IAM 角色的权限。SCP 不授予权限。
- Amazon Organizations RCP：使用 Amazon Organizations 资源控制策略（RCP）为组织或组织单元（OU）中账户内的资源定义最大权限。RCP 限制了基于身份和基于资源的策略可以向组织内账户中的资源授予的权限。RCP 不授予权限。
- 访问控制列表 (ACL) - 使用 ACL 来控制其他账户中的哪些主体可以访问 ACL 附加到的资源。ACL 类似于基于资源的策略，但它们是唯一不使用 JSON 策略文档结构的策略类型。ACL 是跨账户的权限策略，向指定的主体授予权限。ACL 不能向同一账户内的实体授予权限。
- AmazonRAM 资源共享 — 使用 Amazon Resource Access Manager 在 Amazon Web Services 账户、组织单元或整个组织之间共享资源，而无需为每个共享资源编写基于资源的单独策略。
- 会话策略 - 在您使用 Amazon CLI 或 Amazon API 担任某个角色或联合身份用户时，传递高级会话策略。会话策略限制角色或用户的基于身份的策略授予会话的权限。会话策略限制所创建会话的权限，但不授予权限。有关更多信息，请参阅会话策略。

基于身份的策略

基于身份的策略是 JSON 权限的策略文档，用于控制身份（用户、用户组和角色）可在什么样的条件下对哪些资源执行哪些操作。基于身份的策略可以进一步分类：

- 托管策略 – 基于身份的独立策略，可附加到您的 Amazon Web Services 账户 中的多个用户、组和角色。有两种托管策略：

- Amazon 托管策略 - 由 Amazon 创建和管理的托管策略。
- 客户管理型策略 – 您在 Amazon Web Services 账户 中创建和管理的管理型策略。与 Amazon 托管策略相比，客户托管策略可以更精确地控制策略。

- 内联策略 — 直接添加到单个用户、组或角色的策略。内联策略维持策略与身份之间严格的一对一关系。当您删除身份时，它们将会被删除。

要了解如何在托管策略或内联策略之间选择，请参阅 在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源（如 Amazon S3 存储桶）的 JSON 策略文档。这些策略授予指定的主体对该资源执行特定操作的权限，并定义这在哪些条件下适用。基于资源的策略是内联策略。没有基于托管资源的策略。

要启用跨账户访问，您可以将整个账户或其它账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当主体和资源位于单独的 Amazon Web Services 账户 中时，还必须使用基于身份的策略来授予对资源的主体访问权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关授予跨服务访问权限的分步说明，请参阅 IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限。

IAM 服务仅支持一种类型的基于资源的策略（称为角色信任策略），这种策略附加到 IAM 角色。IAM 角色既是身份，又是支持基于资源的策略的资源。因此，您必须将信任策略和基于身份的策略同时附加到 IAM 角色。信任策略定义哪些主体实体（账户、用户、角色和联合身份用户）可以代入该角色。要了解 IAM 角色如何与其他基于资源的策略不同，请参阅 IAM 中的跨账户资源访问。

要了解哪些其他服务支持基于资源的策略，请参阅使用 IAM 的 Amazon 服务。要了解基于资源的策略的更多信息，请参阅 基于身份的策略和基于资源的策略。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

VPC 端点策略

端点策略是一种基于资源的策略，您可以将其附加到 VPC 端点，以控制哪些主体可使用端点，哪些资源可通过端点进行访问。VPC 端点策略不会覆盖或取代附加到目标服务的基于身份的策略或基于资源的策略，VPC 端点策略作为额外的访问边界，其范围仅作用于通过端点的流量。VPC 端点策略用于确保只有 Amazon Organizations 组织中的可信身份，才能通过您的 VPC 端点来访问可信资源。如果您不附加自定义端点策略，则 Amazon 将附加一个允许完全访问的默认策略。有关端点策略的更多信息，请参阅《Amazon PrivateLink 指南》中的使用端点策略控制对 VPC 端点的访问。

IAM 权限边界

权限边界是一项高级功能，借助该功能，您可以设置基于身份的策略可以授予 IAM 实体的最大权限。当您设置实体的权限边界时，该实体只能执行其基于身份的策略和其权限边界同时允许的操作。如果您在基于资源的策略的主体元素中指定角色会话或用户，则无需在权限边界内显式允许。但是，如果您在基于资源的策略的主体元素中指定角色 ARN，则需要在权限边界内显式允许。在这两种情况下，在权限边界内显式拒绝都是有效的。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 实体的权限边界。

Amazon Organizations 服务控制策略（SCP）

如果在组织内启用了所有特征，则可对任意或全部账户应用服务控制策略（SCP）。SCP 是为组织或组织单元 (OU) 的账户内的 IAM 用户和 IAM 角色指定最大权限的 JSON 策略。SCP 限制成员账户中主体（包括每个 Amazon Web Services 账户根用户）的权限。任一项策略中的显式拒绝将覆盖其他策略中的允许。

有关 Amazon Organizations 和 SCP 的更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略 (SCP)。

Amazon Organizations 资源控制策略（RCP）

如果在组织内启用了所有功能，则可使用资源控制策略（RCP）在多个 Amazon Web Services 账户内对资源集中应用访问控制。RCP 是 JSON 策略，您可以使用它们设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制了成员账户中资源的权限，并可能影响身份（包括 Amazon Web Services 账户根用户）的有效权限，无论这些身份是否属于您的组织。任何适用的 RCP 中的显式拒绝将覆盖可能附加到个人身份或资源的其他策略中的允许。

有关 Amazon Organizations 和 RCP（包括支持 RCP 的 Amazon Web Services 服务 列表）的更多信息，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCP)。

访问控制列表（ACL）

访问控制列表 (ACL) 是一种服务策略，允许您控制另一个账户中的哪些主体可以访问资源。不能使用 ACL 控制同一账户中的主体的访问权限。ACL 类似于基于资源的策略，但它们是唯一不使用 JSON 策略文档格式的策略类型。Amazon S3、Amazon WAF 和 Amazon VPC 是支持 ACL 的服务示例。要了解有关 ACL 的更多信息，请参阅《Amazon Simple Storage Service 开发人员指南》中的访问控制列表（ACL）概述。

Amazon Resource Access Manager (Amazon RAM) 资源共享

Amazon Resource Access Manager (Amazon RAM) 允许您与其他 Amazon Web Services 账户、组织单元或 Amazon Organizations 的整个组织共享您在一个 Amazon Web Services 账户 中创建的资源，而无需为每个共享资源编写基于资源的单独策略。虽然您可以通过直接将基于资源的策略附加到资源来授予跨账户访问权限，但 Amazon RAM 提供了一种托管的集中式替代方案，从而无需在策略中逐一列出账户 ID，也无需跨账户维护相同的策略文档。借助 RAM，使用账户可在其服务控制台中以原生方式查看共享资源，资源所有者保留对资源的完全所有权和访问对象的可见性，托管权限则定义了使用者可执行的最大操作范围，这一切操作都由单个资源共享，而无需对每个资源策略进行单独设置。常见使用案例包括跨账户共享 VPC 子网、Amazon 中转网关连接、Route 53 解析器规则和 License Manager 配置。有关更多信息，请参阅《Amazon RAM 用户指南》中的什么是 Amazon Resource Access Manager。

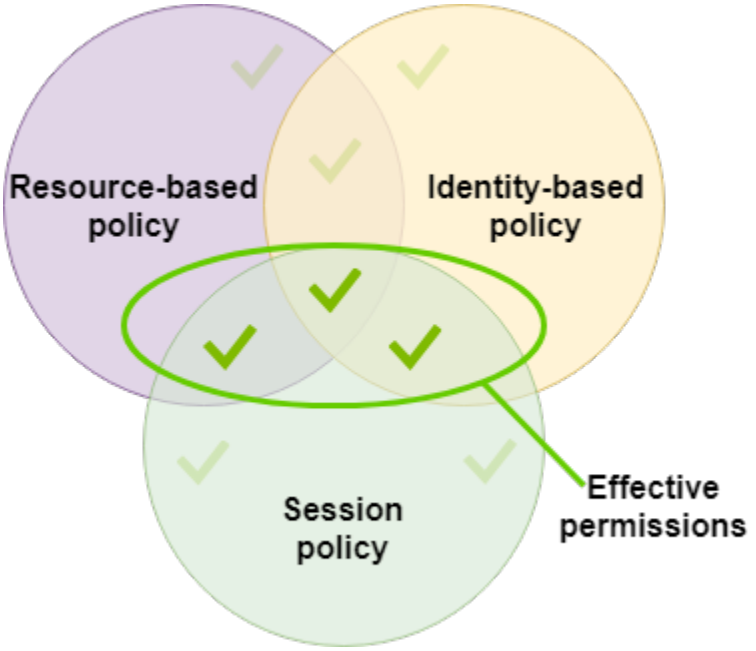
会话策略

会话策略是当您以编程方式为角色或 Amazon STS 联合用户主体创建临时会话时作为参数传递的高级策略。会话的权限是用于创建会话的 IAM 实体（用户或角色）的基于身份的策略与会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。

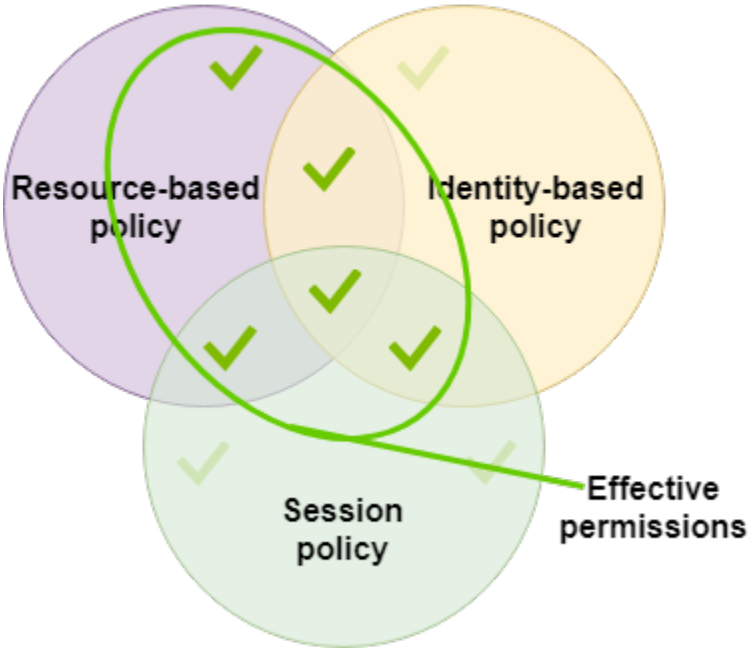
您可以使用 `AssumeRole`、`AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` API 操作以编程方式创建角色会话和传递会话策略。您可以使用 `Policy` 参数传递单个 JSON 内联会话策略文档。您可以使用 `PolicyArns` 参数指定最多 10 个托管会话策略。有关创建角色会话的更多信息，请参阅 临时安全凭证的权限。

当您创建 Amazon STS 联合用户主体会话时，您使用 IAM 用户的访问密钥以编程方式调用 `GetFederationToken` API 操作。您还必须传递会话策略。生成的会话的权限是基于身份的策略与会话策略的交集。有关创建联合身份用户的更多信息，请参阅 通过自定义身份凭证代理程序请求凭证。

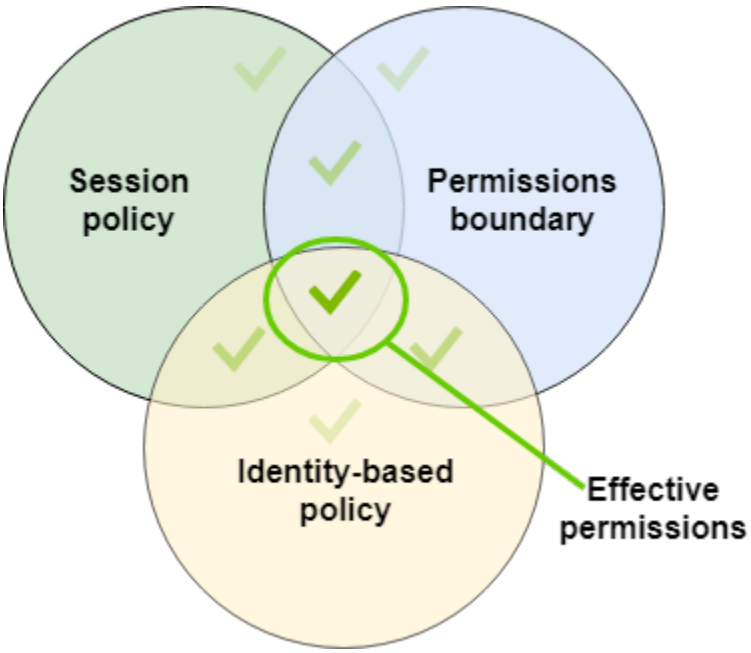
基于资源的策略可以将用户或角色的 ARN 指定为主体。在这种情况下，在创建会话之前，将在角色或用户的基于身份的策略中添加基于资源的策略中的权限。会话策略限制由基于资源的策略和基于身份的策略授予的总权限。生成的会话权限是会话策略与基于资源的策略的交集以及会话策略与基于身份的策略的交集。



基于资源的策略可以将会话的 ARN 指定为主体。在这种情况下，在创建会话后，将添加基于资源的策略中的权限。基于资源的策略权限不受会话策略限制。生成的会话具有基于资源的策略中的所有权限以及 基于身份的策略与会话策略的交集。



权限边界可以设置用于创建会话的用户或角色的最大权限。在这种情况下，生成的会话的权限是会话策略、权限边界和基于身份的策略的交集。不过，权限边界不会限制指定生成的会话 ARN 基于资源的策略授予权限。



策略和根用户

Amazon Web Services 账户根用户 会受到一些策略类型的影响，但不会受其他策略类型的影响。您不能将基于身份的策略附加到根用户，也不能为根用户设置权限边界。不过，您可以在基于资源的策略或 ACL 中将根用户指定为主体。根用户仍然是账户的成员。如果该账户是 Amazon Organizations 中的组成成员，则根用户受账户的 SCP 和 RCP 影响。

JSON 策略概述

大多数策略在 Amazon 中存储为 JSON 文档。基于身份的策略和用于设置权限边界的策略是您附加到用户或角色的 JSON 策略文档。基于资源的策略是附加到资源的 JSON 策略文档。SCP 和 RCP 是附加到 Amazon Organizations 的组织根、组织单元 (OU) 或账户的使用限制语法的 JSON 策略文档。ACL 也可附加到资源，但必须使用不同的语法。会话策略是您在创建角色或联合身份用户会话时提供的 JSON 策略。

您无需了解 JSON 语法。您可以使用 Amazon Web Services 管理控制台中的可视化编辑器创建和编辑客户托管策略，而无需使用 JSON。不过，如果在组中使用内联策略或复杂的策略，您还必须使用控制台在 JSON 编辑器中创建和编辑这些策略。有关使用可视化编辑器的更多信息，请参阅使用客户管理型策略定义自定义 IAM 权限和编辑 IAM 策略。

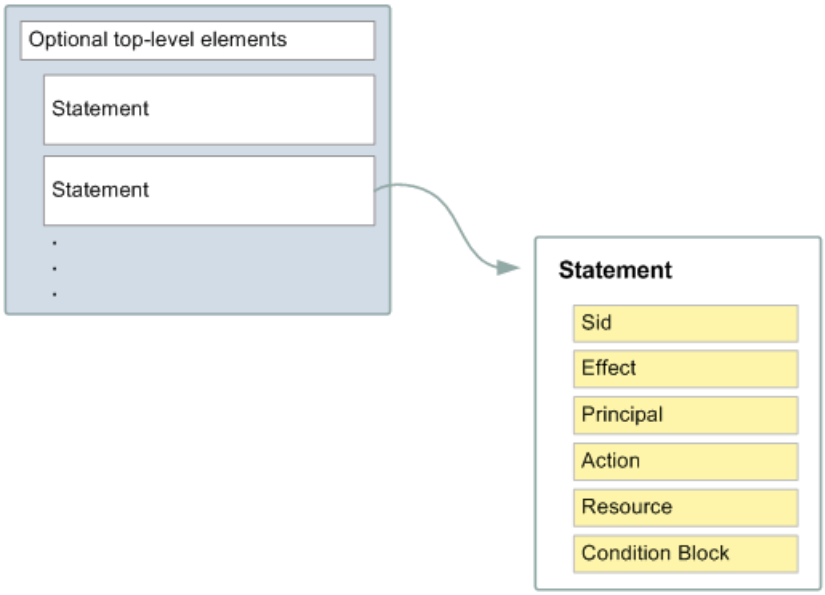
当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

JSON 策略文档结构

如下图所示，JSON 策略文档包含以下元素：

- 文档顶部的可选策略范围信息
- 一个或多个单独语句

每个语句都包含有关单个权限的信息。如果一个策略包含多个语句，则 Amazon 会在评估它们时跨这些语句应用逻辑 OR。如果有多个策略应用于请求，则 Amazon 会在评估它们时跨所有这些策略应用逻辑 OR。



语句中的信息均含在一系列的元素内。

- Version - 指定要使用的策略语言版本。建议您使用最新的 2012-10-17 版本。有关更多信息，请参阅 IAM JSON 策略元素：Version。
- Statement - 将该主要策略元素作为以下元素的容器。可以在一个策略中包含多个语句。
- Sid（可选）- 包括可选的语句 ID 以区分不同的语句。
- Effect - 使用 Allow 或 Deny 指示策略是允许还是拒绝访问。
- Principal（在某些情况下需要）- 如果创建基于资源的策略，则必须指示要允许或拒绝访问的账户、用户、角色或 Amazon STS 联合用户主体。如果要创建 IAM 权限策略以附加到用户或角色，则不能包含该元素。主体暗示为该用户或角色。
- Action - 包括策略允许或拒绝的操作列表。

- Resource（在某些情况下需要）- 如果创建 IAM 权限策略，则必须指定操作适用的资源列表。如果创建基于资源的策略，则是否需要此元素取决于您使用的资源。
- Condition（可选）- 指定策略在哪些情况下授予权限。

要了解上述及其他更高级的策略元素，请参阅IAM JSON 策略元素参考。

多个声明和多个策略

如果要为实体（用户或角色）定义多个权限，您可以在单个策略中使用多个语句。您也可以附加多个策略。如果您尝试在单个语句中定义多个权限，则策略可能没有授予预期访问权限。建议您按资源类型分解策略。

由于策略的大小有限，可能需要对更复杂的权限使用多个策略。在单独的客户管理型策略中创建权限的功能分组也是个好主意。例如，为 IAM 用户管理创建一个策略，为自我管理创建一个策略，并为 S3 存储桶管理创建另一个策略。无论多个语句和多个策略如何组合，Amazon 都会以相同方式评估您的策略。

例如，以下策略具有三个语句，其中每个语句在单个账户中定义一组单独的权限。这些语句定义以下权限：

- `Sid`（语句 ID）为 `FirstStatement` 的第一个语句让具有附加策略的用户更改自己的密码。该语句中的 `Resource` 元素是“*”（这表示“所有资源”）。但实际上，`ChangePassword` API 操作（或等效的 `change-password` CLI 命令）仅影响发出请求的用户的密码。
- 第二个语句使用户可以列出其 Amazon Web Services 账户 中的所有 Amazon S3 存储桶。该语句中的 `Resource` 元素是 `"*"`（这表示“所有资源”）。但由于策略没有为其他账户中的资源授予访问权限，因此，用户只能列出自己的 Amazon Web Services 账户 中的存储桶。
- 第三个语句允许用户列出和检索名为 `amzn-s3-demo-bucket-confidential-data` 的存储桶中的任何对象，但是仅当使用 多重身份验证 (MFA) 对用户进行了身份验证时才能如此。策略中的 `Condition` 元素将强制实施 MFA 身份验证。

  如果策略语句包含 `Condition` 元素，则仅当 `Condition` 元素计算为 true 时，语句才有效。在此示例中，`Condition` 在用户进行了 MFA 身份验证时计算为 true。如果用户没有进行 MFA 身份验证，则此 `Condition` 计算为 false。在这种情况下，此策略中的第三个语句不会应用，因此用户将无法访问 `amzn-s3-demo-bucket-confidential-data` 存储桶。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "FirstStatement",
      "Effect": "Allow",
      "Action": ["iam:ChangePassword"],
      "Resource": "*"
    },
    {
      "Sid": "SecondStatement",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    },
    {
      "Sid": "ThirdStatement",
      "Effect": "Allow",
      "Action": [
        "s3:List*",
        "s3:Get*"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket-confidential-data",
        "arn:aws:s3:::amzn-s3-demo-bucket-confidential-data/*"
      ],
      "Condition": {"Bool": {"aws:MultiFactorAuthPresent": "true"}}
    }
  ]
}
```

JSON 策略语法示例

以下基于身份的策略允许暗示的主体列出名为 amzn-s3-demo-bucket 的单个 Amazon S3 存储桶：

JSON

```
{
```

```
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket"
  }
}
```

以下基于资源的策略可附加到 Amazon S3 存储桶。此策略允许特定 Amazon Web Services 账户 的成员在名为 amzn-s3-demo-bucket 的存储桶中执行任何 Amazon S3 操作。它允许可对存储桶或其中的对象执行的任何操作。(因为该策略仅向该账户授予信任，所以仍必须向该账户中的各个用户授予执行指定 Amazon S3 操作的权限。)

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "1",
            "Effect": "Allow",
            "Principal": {
                "AWS": [
                    "arn:aws:iam::111122223333:root"
                ]
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

要查看常见方案的示例策略，请参阅IAM 基于身份的策略示例。

授予最低权限

创建 IAM 策略时，请遵循授予最低权限这一标准安全建议，或仅授予执行任务所需的权限。确定用户和角色需要执行的操作，然后制订允许他们仅执行这些任务的策略。

最开始只授予最低权限，然后根据需要授予其它权限。这样做比起一开始就授予过于宽松的权限而后再尝试收紧权限来说更为安全。

作为最低权限的替代方案，您可以使用 Amazon 托管策略 或带通配符 * 权限的策略开始使用策略。考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。监控这些主体以了解他们正在使用哪些权限。然后写入最低权限策略。

IAM 提供了多个选项来帮助您优化授予的权限。

- 了解访问级别分组 - 您可以使用访问级别分组来了解策略授予的访问级别。策略操作被归类为 `List`、`Read`、`Write`、`Permissions management` 或 `Tagging`。例如，您可以从 `List` 和 `Read` 访问级别中选择操作，以向您的用户授予只读访问权限。要了解如何使用策略摘要来了解访问级别权限，请参阅 策略摘要中的访问级别。
- 验证您的策略 — 您可以在创建和编辑 JSON 策略时使用 IAM Access Analyzer 执行策略验证。我们建议您查看和验证所有现有策略。IAM Access Analyzer 提供 100 多项策略检查来验证您的策略。当您策略中的语句允许我们认为过于宽容的访问时，它会生成安全警告。在授予最低权限时，您可以使用通过安全警告提供的可操作建议。要了解有关 IAM Access Analyzer 提供的策略检查的更多信息，请参阅 IAM Access Analyzer 策略验证。
- 基于访问活动生成策略 — 为了帮助您优化授予的权限，您可以根据 IAM 实体（用户或角色）的访问活动生成 IAM 策略。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含实体在指定时间框架内使用的权限。您可以使用模板创建具有精细权限的托管策略，然后将其附加到 IAM 实体。这样，您仅需授予用户或角色与特定使用案例中的 Amazon 资源进行交互所需的权限。有关更多信息，请参阅 IAM Acess Analyzer 策略生成。
- 使用上次访问的信息 — 另一项可帮助提供最低权限的功能是上次访问的信息。可以在 IAM 控制台详细信息页面上的访问顾问选项卡中查看此信息，以了解 IAM 用户、组、角色或策略。上次访问的信息还包括有关上次访问某些服务（如 Amazon EC2、IAM、Lambda 和 Amazon S3）的操作的信息。如果您使用 Amazon Organizations 管理账户凭证登录，则可以在 IAM 控制台的 Amazon Organizations 部分查看上次访问的服务信息。也可以使用 Amazon CLI 或 Amazon API 为 IAM 或 Amazon Organizations 中的实体或策略检索上次访问的信息报告。您可以使用此信息确定不必要的权限，从而优化 IAM 或 Amazon Organizations 策略以更好地遵循最低权限原则。有关更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。
- 查看 Amazon CloudTrail 中的账户事件 — 要进一步减少权限，您可以在 Amazon CloudTrail Event history（事件历史记录）中查看您的账户事件。CloudTrail 事件日志包含详细的事件信息，您可以用

来减少策略的权限。这些日志仅包含 IAM 实体所需的操作和资源。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的在 CloudTrail 控制台中查看 CloudTrail 事件。

有关更多信息，请参阅以下单个服务的策略主题，其中提供了如何针对特定产品的资源编写策略的示例。

- 《Amazon DynamoDB 开发人员指南》中的将基于资源的策略用于 DynamoDB
- 《Amazon Simple Storage Service 用户指南》中的 Amazon S3 的存储桶策略
- 《Amazon Simple Storage Service 用户指南》中的访问控制列表（ACL）概述

托管策略与内联策略

如果在 IAM 中为身份设置权限，您必须决定是使用 Amazon 托管式策略、客户管理型策略还是内联策略。以下主题更详细说明了每种基于身份的策略以及何时使用这些策略。

下表概述了这些策略：

策略类型	说明	谁管理策略？	修改权限？	应用于策略的主体数量？
Amazon托管策略	由 Amazon 创建和管理的独立策略。	Amazon	否	很多
客户托管策略	为特定使用案例创建的策略，并且可以根据需要随时更改或更新它们。	You	是	很多
内联策略	为单个 IAM 身份（用户、组或角色）创建的策略，该策略在策略和身份之间保	You	是	One

策略类型	说明	谁管理策略？	修改权限？	应用于策略的主体数量？
	持严格的一对一关系。			

主题

- Amazon托管策略
- 客户托管策略
- 内联策略
- 在托管策略与内联策略之间进行选择
- 将内联策略转换为托管式策略
- 已弃用的 Amazon 托管策略

Amazon托管策略

Amazon 托管策略 是由 Amazon 创建和管理的独立策略。独立策略意味着策略有自身的 Amazon 资源名称 (ARN)，其中包含策略名称。例如，`arn:aws:iam::aws:policy/IAMReadOnlyAccess` 是一个 Amazon 托管策略。有关 ARN 的更多信息，请参阅 IAM ARN。有关适用于 Amazon Web Services 服务 的 Amazon 托管式策略的列表，请参阅 Amazon 托管式策略。

借助 Amazon 托管式策略，您可以方便地为用户、IAM 组和角色分配适当的权限。它比自己编写策略更快，并且包括许多常见使用案例的权限。

您不能更改 Amazon 托管策略中定义的权限。Amazon 有时会更新 Amazon 托管策略中定义的权限。在 Amazon 执行此操作时，更新会影响策略附加到的所有主体实体（IAM 用户、IAM 组和 IAM 角色）。在推出新的 Amazon 服务或为现有服务提供新的 API 调用时，Amazon 很可能会更新 Amazon 托管式策略。例如，名为 ReadOnlyAccess 的 Amazon 托管策略提供针对所有 Amazon Web Services 服务 和资源的只读访问权限。在 Amazon 推出新的服务时，Amazon 将更新 ReadOnlyAccess 策略，以便为新服务添加只读权限。更新的权限会应用于策略附加到的所有主体实体。

完全访问 Amazon 托管策略：这些策略通过授予对服务的完全访问权限来定义服务管理员的权限。示例包括：

- AmazonDynamoDBFullAccess
- IAMFullAccess

高级用户 Amazon 托管策略：这些策略提供对 Amazon Web Services 服务 和资源的完全访问权限，但不允许管理用户和 IAM 组。示例包括：

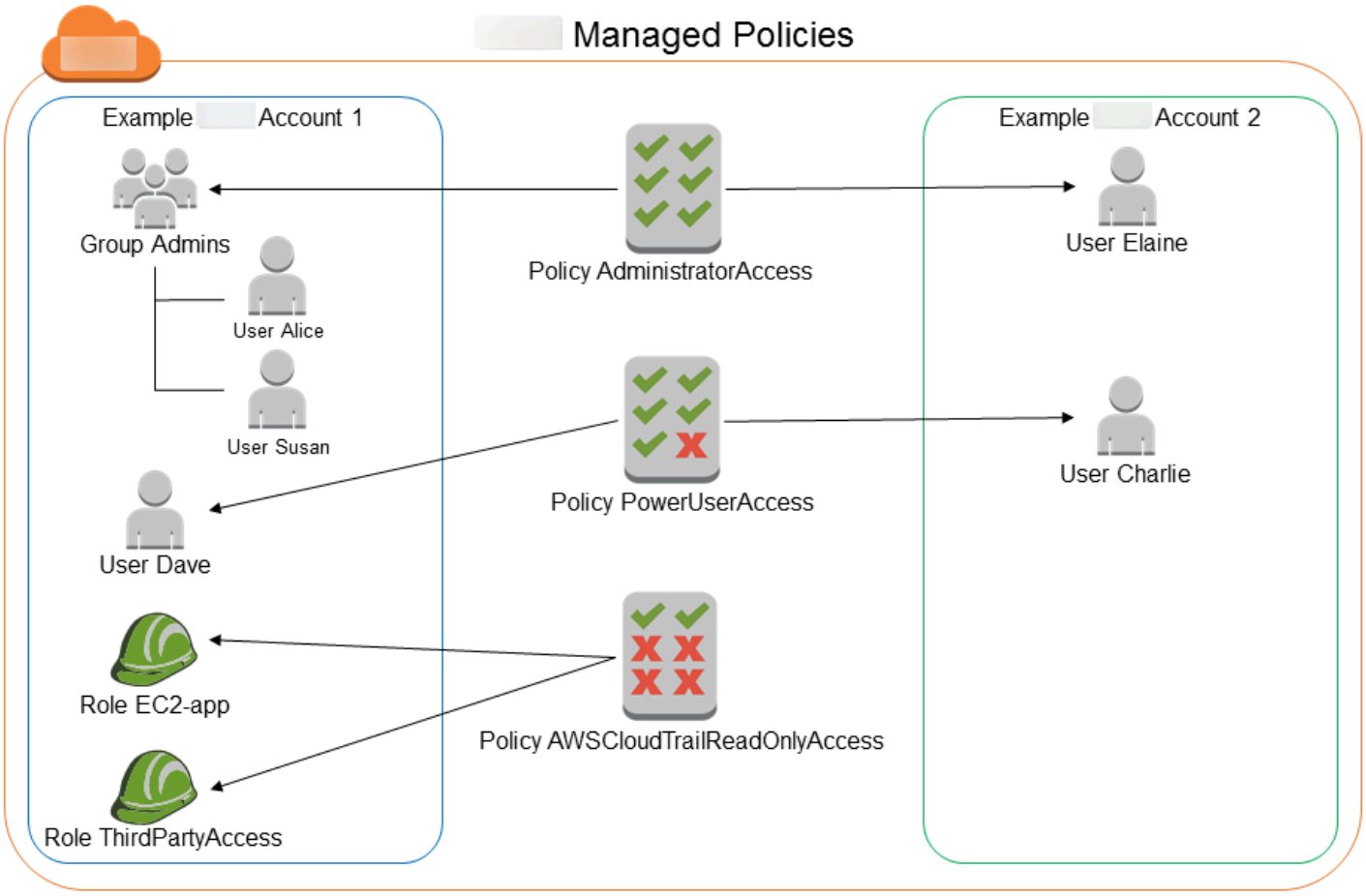
- AWSCodeCommitPowerUser
- AWSKeyManagementServicePowerUser

部分访问 Amazon 托管策略：这些策略提供对 Amazon Web Services 服务 的特定访问级别，但不允许权限管理访问级别权限。示例包括：

- AmazonMobileAnalyticsWriteOnlyAccess
- AmazonEC2ReadOnlyAccess

工作职能 Amazon 托管策略：这些策略与 IT 行业中常用的工作职能密切相关，便于为这些工作职能授予权限。使用工作职能策略的一个重要优势是，在推出新的服务和 API 操作时，Amazon 对这些策略进行维护和更新。例如，AdministratorAccess 工作职能提供对 Amazon 中的每个服务和资源的完全访问权限和权限委派。我们建议您仅对账户管理员使用此策略。对于需要对 IAM 和 Amazon Organizations 的有限访问权限以及此外所有服务的完全访问权限的高级用户，请使用 PowerUserAccess 工作职能。有关工作职能策略的列表和说明，请参阅工作职能的 Amazon 托管策略。

下图对 Amazon 托管策略进行说明。该图显示三个 Amazon 托管式策略：AdministratorAccess、PowerUserAccess 和 AmazonCloudTrail_ReadOnlyAccess。请注意，单个 Amazon 托管策略可以附加到不同 Amazon Web Services 账户 中的主体实体，并且可以附加到单个 Amazon Web Services 账户 中的不同主体实体。



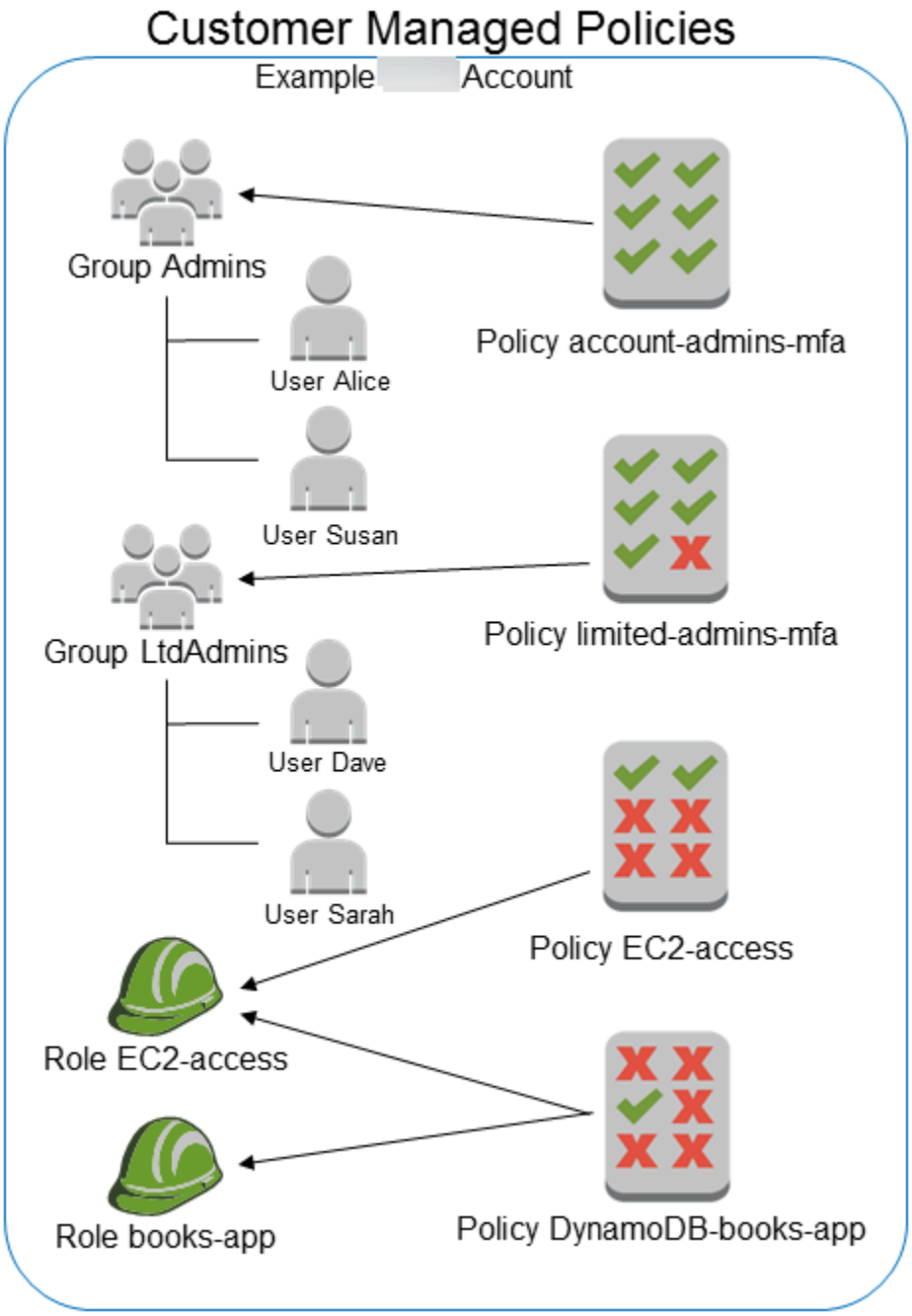
客户托管策略

您可以在自己的 Amazon Web Services 账户 中创建可附加到主体实体（IAM 用户、IAM 组和 IAM 角色）的独立策略。您可以为特定使用案例创建这些客户管理型策略，并且可以根据需要随时更改和更新它们。与 Amazon 托管式策略一样，当您将策略附加到主体实体时，会向实体授予策略中定义的权限。当您更新策略中的权限时，这些更改将应用于策略所附加的所有主体实体。

创建客户管理型策略的理想方式是：首先复制一个现有 Amazon 托管策略。这样从一开始您就可以确信策略是正确的，只需根据您的环境进行自定义即可。

下面的示意图对客户托管策略进行说明。每个策略都是 IAM 中的一个实体，有自己的 Amazon 资源名称 (ARN)，其中包含策略名称。请注意，同一策略可以附加到多个主体实体；例如，同一 DynamoDB-books-app 策略附加到两个不同的 IAM 角色。

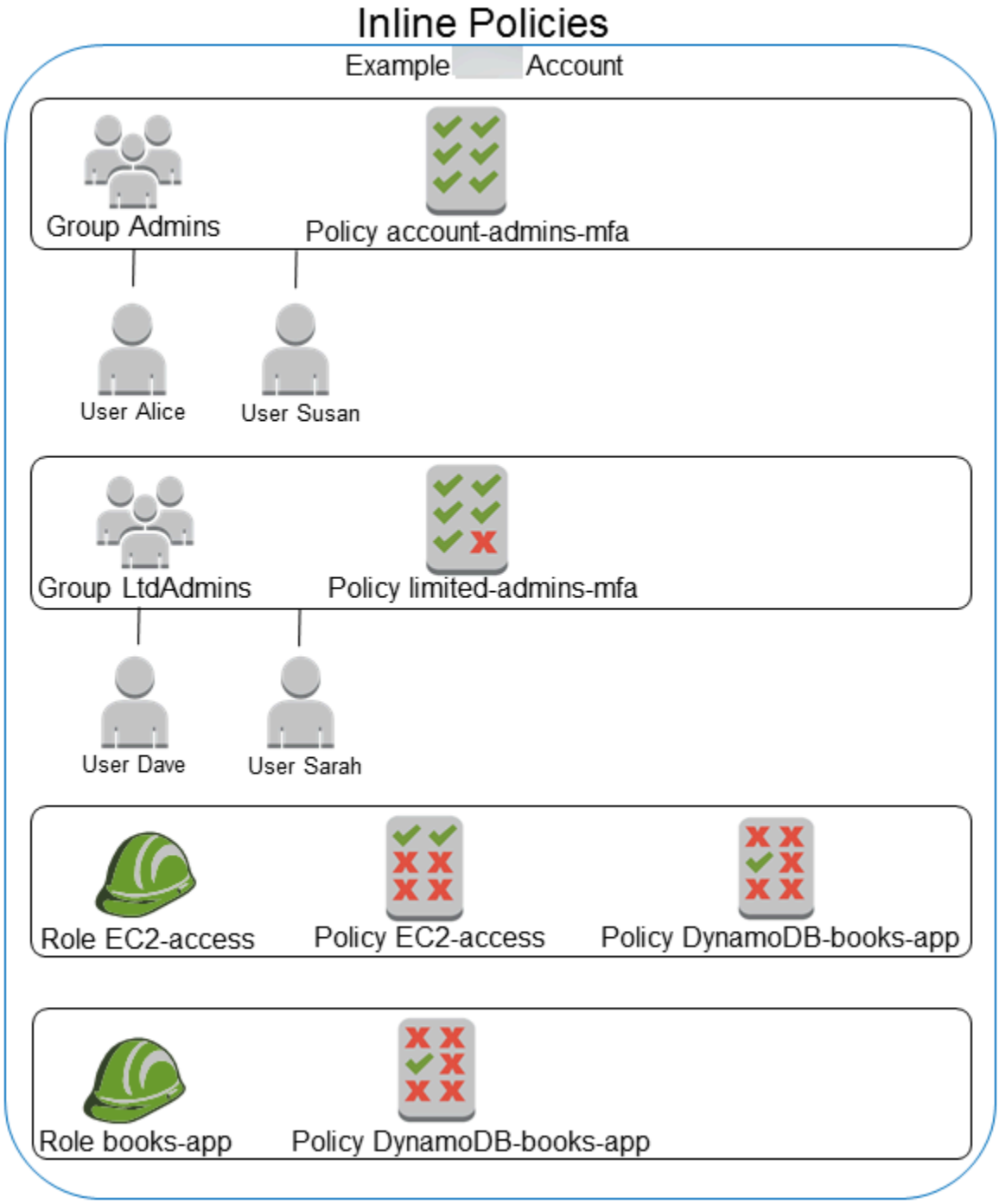
有关更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。



内联策略

内联策略是为单个 IAM 身份（用户、用户组或角色）创建的策略。内联策略维持策略与身份之间严格的一对一关系。当您删除身份时，它们将会被删除。您可以创建策略并将其嵌入到身份中，无论是在创建身份时还是之后。如果一个策略可以应用于多个实体，则最好使用托管式策略。

下面的示意图对内联策略进行说明。每个策略都是用户、组或角色的固有部分。请注意，两个角色包含同一策略（DynamoDB-books-app 策略），但是它们不共享单个策略。每个角色都有自己的策略副本。



在托管策略与内联策略之间进行选择

在决定托管式策略和内联策略时，请考虑您的使用案例。在大多数情况下，我们建议使用托管策略而不是内联策略。

> **Note**
>
> 您可以同时使用托管式策略和内联策略来定义主体实体的通用权限和唯一权限。

托管策略具备以下功能：

可重复使用性

单个托管策略可以附加到多个主体实体 (用户、组和角色)。您可以创建策略库，在库中定义对您的 Amazon Web Services 账户 有用的权限，然后根据需要将这些策略附加到主体实体。

集中变更管理

更改托管策略时，更改会应用于策略附加到的所有主体实体。例如，假设要为新的 Amazon API 添加权限，则可以通过更新客户管理型策略或关联 Amazon 托管式策略来添加权限。如果使用 Amazon 托管式策略，则 Amazon 会更新该策略。更新托管式策略时，更改会应用于附加了该托管式策略的所有主体实体。相比之下，要更改内联策略，则必须分别编辑包含该策略的每个身份。例如，如果一个组和一个角色都包含同一内联策略，则您必须分别编辑这两个主体实体来更改该策略。

版本控制和回滚

在更改客户托管策略时，更改的策略不会覆盖现有的策略。而是由 IAM 创建新的托管策略版本。IAM 最多可以存储五个版本的客户管理策略。您可以根据需要使用策略版本将策略还原为较早版本。

> **Note**
>
> 策略版本与 `Version` 策略元素不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。要了解策略版本的更多信息，请参阅the section called “IAM 策略版本控制”。要了解 `Version` 策略元素的更多信息，请参阅IAM JSON 策略元素：Version。

委派权限管理

您可以允许您 Amazon Web Services 账户 中的用户附加和分离策略，同时保持对这些策略中定义的权限的控制。为此，请将一些用户指定为完全管理员，即可以创建、更新和删除策略的管理员。然后，您可以将其他用户指定为受限管理员。这些受限制的管理员可以将策略附加到其他主体实体，但只能附加您允许他们附加的策略。

有关委派权限管理的更多信息，请参阅控制对策略的访问。

上调策略字符数限制

托管式策略的最大字符数限制大于组内联策略的字符数限制。如果您已达到内联策略的字符数限制，则可以创建更多 IAM 组并将托管式策略附加到该组。

有关限额和限制的更多信息，请参阅 IAM 和 Amazon STS 配额。

Amazon 托管策略的自动更新

Amazon 维护 Amazon 托管式策略并在需要时更新它们，例如，针对新 Amazon 服务添加权限，您不必进行更改。更新会自动应用于已附加了 Amazon 托管策略的主体实体。

开始使用托管式策略

我们建议使用授予最低权限的策略，或仅授予执行任务所需的许可。授予最低权限的最安全方式是编写一个仅具有团队所需权限的客户管理型策略。您必须创建一个流程，以允许您的团队在必要时请求更多权限。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。

要开始向您的 IAM 身份（用户、用户组和角色）添加权限，您可以使用 Amazon托管策略。Amazon托管策略不会授予最低权限。您必须考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。

您可以将 Amazon 托管策略（包括任务函数）附加到任何 IAM 身份。有关更多信息，请参阅 添加和删除 IAM 身份权限。

要切换到最低权限，您可以运行 Amazon Identity and Access Management and Access Analyzer 以使用 Amazon 托管式策略监控主体。了解他们使用的权限后，您可以编写或生成仅包含团队所需权限的客户管理型策略。这中方法不太安全，但您能够以更灵活的方式了解您的团队如何使用 Amazon。有关更多信息，请参阅 IAM Access Analyzer 策略生成。

Amazon 托管策略可用于为很多常用案例提供权限。有关专为特定任务函数制定的 Amazon 托管策略的更多信息，请参阅 工作职能的 Amazon 托管策略。

有关 Amazon 托管式策略的列表，请参阅《Amazon 托管式策略参考指南》。

使用内联策略

如果您要在策略与应用它的身份之间维持严格的一对一关系，则内联策略十分有用。例如，如果您需要确保策略中的权限不会无意中分配给预期身份之外的身份。使用内联策略时，策略中的权限不可能意外分配给错误的身份。此外，当您使用 Amazon Web Services 管理控制台 删除该身份时，嵌入在身份中的策略也会被删除，因为它们是主体实体的一部分。

将内联策略转换为托管式策略

如果您的账户中具有内联策略，则可将其转换为托管策略。为此，请将该策略复制到新的托管策略。接下来，将新策略附加到具有内联策略的身份。然后，请删除内联策略。

将内联策略转换为托管策略

将内联策略转换为托管策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups（组）、Users（用户）或 Roles（角色）。
3. 在列表中，请选择具有要修改的策略的用户组、用户或角色的名称。
4. 选择权限选项卡。
5. 对于 IAM 组，请选择要移除的内联策略的名称。对于用户和角色，选择再显示 ***n*** 个（如有必要），然后展开要删除的内联策略。
6. 选择复制以复制策略的 JSON 策略文档。
7. 在导航窗格中，选择策略。
8. 选择创建策略，然后选择 JSON 选项。
9. 将现有文本替换为您的 JSON 策略文本，然后选择下一步。
10. 输入您策略的名称和可选描述，然后选择创建策略。
11. 在导航窗格中，请选择 Groups（组）、Users（用户）或 Roles（角色），然后再次选择具有要删除的策略的用户组、用户或角色的名称。
12. 请选择权限选项卡，然后选择添加权限。
13. 对于 IAM 组，选中新策略名称旁边的复选框，选择添加权限，然后选择附加策略。对于用户或角色，选择 Add permissions (添加权限)。在下一页上，选择直接附加现有策略，选中您的新策略的名称旁边的复选框，选择下一步，然后选择添加权限。

    您将返回您的用户组、用户或角色的 Summary（摘要）页面。
14. 选中要移除的内联策略旁边的复选框，然后选择移除。

已弃用的 Amazon 托管策略

为了简化权限的分配，Amazon 提供托管策略 — 这是一项预定义策略，可随时附加到您的 IAM 用户、组和角色。

有时 Amazon 需要向现有策略添加新权限，例如在引入新服务时。向现有策略添加新权限不会中断或移除任何功能或能力。

但是，如果所需的更改在应用于现有策略时可能影响客户，则 Amazon 可能会选择创建新 策略。例如，从现有策略中移除权限可能中断依赖该策略的任何 IAM 实体或应用程序的权限，从而可能中断关键操作。

因此，当需要此类更改时，Amazon 会根据所需更改创建一种全新的策略，并将其提供给客户。然后，旧策略被标记为已弃用。有关更多信息，请参阅《Amazon 托管式策略参考指南》中的 Deprecated Amazon managed policies。

使用数据边界建立权限防护机制

数据边界防护机制旨在充当始终开启边界，以帮助保护各种 Amazon 账户和资源中的数据。数据边界遵循 IAM 安全最佳实践在多个账户之间建立权限防护机制。这些组织范围的权限防护机制并不能取代现有的精细访问控制。相反，它们充当粗粒度的访问控制，通过确保用户、角色和资源遵守一组定义的安全标准来帮助改善安全策略。

数据边界是 Amazon 环境中的一组权限防护机制，可帮助确保只有可信身份才能访问来自预期网络的可信资源。

- 可信身份：Amazon 账户中的主体（IAM 角色或用户）以及代表您行事的 Amazon 服务。
- 可信资源：Amazon 账户或代表您行事的 Amazon 服务所拥有的资源。
- 预期网络：本地数据中心和虚拟私有云（VPC）或者代表您行事的 Amazon 服务网络。

> **Note**
>
> 在某些情况下，您可能需要扩展数据边界，以包括可信业务合作伙伴的访问。在创建特定于您的公司和 Amazon Web Services 服务 的使用的可信身份、可信资源和预期网络的定义时，应考虑所有预期的数据访问模式。

应将数据边界控制视为信息安全和风险管理计划中的任何其他安全控制。这意味着您应该执行威胁分析来识别云环境中的潜在风险，然后根据自己的风险接受标准，选择并实施适当的数据边界控制。为了更

好地为基于风险的迭代数据边界实现方法提供信息，您需要了解数据边界控制可以解决哪些安全风险和威胁向量，以及安全优先级。

数据边界控制

数据边界粗粒度控制通过实现不同的 策略类型 和条件键组合，帮助您在三个数据边界上实现六个不同的安全目标。

边界	控制目标	使用	应用于	全局条件上下文键
身份	只有可信身份才能访问我的资源	RCP	资源	aws:PrincipalOrgID
	只有可信身份可以访问您的网络	VPC 端点策略	Network	aws:PrincipalOrgPaths aws:PrincipalAccount aws:PrincipalIsAwsService aws:SourceOrgID aws:SourceOrgPath aws:SourceAccount
资源	您的身份只能访问可信资源	SCP	身份	aws:ResourceOrgID
	只能从您的网络访问可信资源	VPC 端点策略	Network	aws:ResourceOrgPaths aws:ResourceAccount

边界	控制目标	使用	应用于	全局条件上下文键
Network	您的身份只能访问预期网络中的资源	SCP	身份	aws:SourceIp aws:SourceVpc aws:SourceVpce aws:VpceAccount aws:VpceOrgPaths aws:VpceOrgID aws:ViaAWSService aws:PrincipalIsAwsService
	只能从预期网络访问您的资源	RCP	资源	

可以将数据边界视为在数据周围创建牢固的边界，以防止意外访问模式。尽管数据边界可防止广泛的意外访问，但您仍然需要做出精细的访问控制决策。建立数据边界并不能减少使用诸如 IAM Access Analyzer 之类的工具持续微调权限的必要性，这是您实现最低权限的旅程的一部分。

要对 RCP 当前不支持的资源执行数据边界控制，您可以使用直接附加到资源的基于资源的策略。有关支持 RCP 和基于资源的策略的服务列表，请参阅 Resource control policies (RCPs) 和 使用 IAM 的 Amazon 服务。

要强制实施网络边界控制，我们建议使用 `aws:VpceOrgID`、`aws:VpceOrgPaths` 和 `aws:VpceAccount`，但唯一前提是当前支持要限制其访问的所有服务。将条件键用于不支持的服务可能会导致意想不到的授权结果。有关支持这些键的服务的列表，请参阅Amazon 全局条件上下文密钥。如果需要对更广泛的服务强制实施控制，可考虑改用 `aws:SourceVpc` 和 `aws:SourceVpce`。

身份边界

身份边界是一组粗粒度的预防性访问控制，有助于确保只有可信身份才能访问您的资源，并且只有可信身份才能访问您的网络。可信身份通常包括 Amazon 账户中的主体（角色或用户）以及代表您行事的 Amazon 服务。除非授予显式例外，否则所有其他身份都被视为不可信，并且会被身份边界阻止。

根据您对可信身份的定义，以下全局条件键可帮助强制实施身份边界控制。在资源控制策略中使用这些键来限制对资源的访问，或者在 VPC 端点策略中使用这些键来限制对您的网络的访问。

您拥有的身份

可使用以下条件键，定义您在 Amazon Web Services 账户 中创建和管理的 IAM 主体。

- aws:PrincipalOrgID – 可以使用此条件键来确保提出请求的 IAM 主体属于 Amazon Organizations 中的指定组织。
- aws:PrincipalOrgPaths – 可以使用此条件键来确保提出请求的 IAM 用户、IAM 角色、Amazon STS 联合用户主体或 SAML 联合主体、OIDC 联合主体或 Amazon Web Services 账户根用户 属于 Amazon Organizations 中的指定组织单位（OU）。
- aws:PrincipalAccount – 可以使用此条件键来确保只有您在策略中指定的主体账户才能访问资源。

代表您行事的 Amazon 服务的身份

可使用以下条件键，允许 Amazon 服务在代表您行事时使用其自有身份访问您的资源。

- aws:PrincipalIsAWSService 和 aws:SourceOrgID（或者 aws:SourceOrgPaths 和 aws:SourceAccount）– 可以使用这些条件键来确保当 Amazon Web Services 服务 主体访问您的资源时，他们仅代表指定组织、组织单位中的资源或 Amazon Organizations 中的账户进行访问。

有关更多信息，请参阅在 Amazon 上建立数据边界：仅允许可信身份访问公司数据。

资源边界

资源边界是一组粗粒度的预防性访问控制，有助于确保您的身份只能访问可信资源，并且只能从您的网络访问可信资源。可信资源通常包括您的 Amazon 账户或代表您行事的 Amazon 服务所拥有的资源。

根据您对可信资源的定义，以下全局条件键有助于强制实施资源边界控制。在服务控制策略 (SCP)中使用这些键来限制您的身份可以访问哪些资源，或者在 VPC 端点策略中使用这些键来限制可以从您的网络访问哪些资源。

您拥有的资源

可使用以下条件键，定义您在 Amazon Web Services 账户 中创建和管理的 Amazon 资源。

- aws:ResourceOrgID – 可以使用此条件键来确保正在访问的资源属于 Amazon Organizations 中的指定组织。
- aws:ResourceOrgPaths – 可以使用此条件键来确保正在访问的资源属于 Amazon Organizations 中指定的组织单位（OU）。
- aws:ResourceAccount – 可以使用此条件键来确保正在访问的资源属于指定 Amazon Web Services 账户。

代表您行事的 Amazon 服务资源

在某些情况下，您可能需要允许访问 Amazon 拥有的资源，这些资源不属于您的组织，由您的主体或代表您行事的 Amazon 服务访问。有关这些场景的更多信息，请参阅在 Amazon 上建立数据边界：仅允许来自我的组织的可信资源。

网络边界

网络边界是一组粗粒度的预防性访问控制，可帮助确保您的身份只能访问预期网络中的资源，并且只能从预期网络访问您的资源。预期网络通常包括您的本地数据中心和虚拟私有云（VPC）以及代表您行事的 Amazon 服务网络。

根据您对预期网络的定义，以下全局条件键有助于强制实施网络边界控制。在服务控制策略（SCP）中使用这些键来限制您的身份可以与之通信的网络，或者在资源控制策略（RCP）中使用这些键来限制对预期网络的资源访问。

您拥有的网络

可使用以下条件键，定义您的员工和应用程序访问资源时应使用的网络，例如企业 IP CIDR 范围和 VPC。

- aws:SourceIp – 可以使用此条件键来确保请求者的 IP 地址在指定的 IP 范围内。
- aws:SourceVpc – 可以使用此条件键来确保请求通过其传输的 VPC 端点属于指定的 VPC。
- aws:SourceVpce – 可以使用此条件键来确保请求通过指定的 VPC 端点传输。
- aws:VpceAccount – 可以使用此条件键来确保请求经由指定 Amazon 账户拥有的 VPC 端点。
- aws:VpceOrgPaths：可使用此条件键来确保请求经由属于 Amazon Organizations 中指定组织单元（OU）中的账户拥有的 VPC 端点。

- aws:VpceOrgID – 可使用此条件键来确保请求经由 Amazon Organizations 中的指定组织中的账户拥有的 VPC 端点。

`aws:VpceAccount`、`aws:VpceOrgPaths` 和 `aws:VpceOrgID` 对实施网络边界控制特别有用，这种控制可随 VPC 端点的使用情况自动扩展，且在创建新端点时无需更新策略。有关支持这些键的 Amazon Web Services 服务 列表，请参阅Amazon 全局条件上下文密钥。

代表您行事的 Amazon 服务网络

可使用以下条件键，允许 Amazon 服务在代表您行事时从其网络访问您的资源。

- aws:ViaAWSService – 可以使用此条件键来确保 Amazon Web Services 服务 可以使用 转发访问会话（FAS）代表您的主体提出请求。
- aws:PrincipalIsAWSService – 可以使用此条件键来确保 Amazon Web Services 服务 可以使用 Amazon 服务主体 访问您的资源。

在其他情况下，您需要允许从网络外部访问 Amazon Web Services 服务 以访问您的资源。有关更多信息，请参阅在 Amazon 上建立数据边界：仅允许从预期网络访问公司数据。

用于了解有关数据边界的更多信息的资源

以下资源可帮助了解有关 Amazon 的数据边界的更多信息。

- Amazon 上的数据边界 – 了解数据边界及其优势和用例。
- 博客文章系列：在 Amazon 上建立数据边界 – 这些博客文章涵盖了有关大规模建立数据边界的规范性指导，包括关键的安全和实现注意事项。
- 数据边界策略示例 – 此 GitHub 存储库包含示例策略，这些策略涵盖了一些常见模式，可帮助您在 Amazon 上实现数据边界。
- 数据边界帮助程序 – 此工具通过分析 Amazon CloudTrail 日志中的访问活动，帮助您设计和预测数据边界控制的影响。

IAM 实体的权限边界

Amazon 对于 IAM 实体（用户或角色）支持权限边界。权限边界是一个高级功能，它使用托管策略设置基于身份的策略可以为 IAM 实体授予的最大权限。实体的权限边界仅允许实体执行其基于身份的策略和权限边界同时允许的操作。

有关策略类型的更多信息，请参阅策略类型。

> **Important**
>
> 如果基于资源的策略语句包含对附加了权限边界策略的 IAM 用户或角色具有 Deny 效果的 NotPrincipal 策略元素，则不要使用这种策略语句。具有 Deny 效果的 NotPrincipal 元素将始终拒绝任何附加了权限边界策略的 IAM 主体，无论 NotPrincipal 元素中指定的值为何。这会导致本来可以访问该资源的某些 IAM 用户或角色失去访问权限。我们建议更改基于资源的策略语句，以将 ArnNotEquals 条件运算符和 aws:PrincipalArn 上下文键结合使用来限制访问权限，而不是使用 NotPrincipal 元素。有关 NotPrincipal 元素的信息，请参阅 Amazon JSON 策略元素：NotPrincipal。

您可以使用 Amazon 托管策略或客户管理型策略为 IAM 实体（用户或角色）设置边界。该策略限制用户或角色的最大权限。

例如，假设名为 Shirley 的 IAM 用户应仅允许管理 Amazon S3、Amazon CloudWatch 和 Amazon EC2。要执行此规则，您可以使用以下策略为 Shirley 用户设置权限边界：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        }
    ]
}
```

当使用策略为用户设置权限边界时，它会限制用户的权限，但自己不提供权限。在本示例中，策略设置 Shirley 的最大权限作为 Amazon S3、CloudWatch 和 Amazon EC2 中的所有操作。Shirley 无法在任何其他服务（包括 IAM）中执行操作，即使她有一个允许这样做的权限策略也是如此。例如，您可以将以下策略添加给 Shirley 用户：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

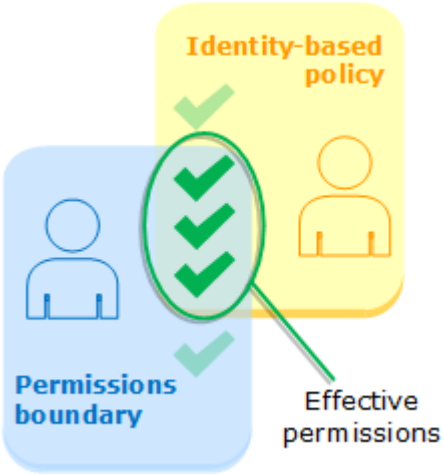
此策略允许在 IAM 中创建用户。如果您将此权限策略附加到 Shirley 用户，然后 Shirley 尝试创建用户，该操作会失败。它由于权限边界不允许执行 `iam:CreateUser` 操作而失败。鉴于这两个策略，Shirley 无权在 Amazon 中执行任何操作。您必须添加不同的权限策略以允许其他服务中的操作，例如 Amazon S3。或者，您也可以更新权限边界，以允许她在 IAM 中创建用户。

评估具有边界的有效权限

IAM 实体（用户或角色）的权限边界设置实体可以具有的最大权限。这可以更改该用户或角色的有效权限。实体的有效权限是影响用户或角色的所有策略所授予的权限。在账户中，基于身份的策略、基于资源的策略、权限边界、Amazon Organizations SCP 或会话策略可以影响实体的权限。有关不同类型的策略的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

如果以下任一策略类型显式拒绝操作的访问权限，则请求会被拒绝。多个权限类型向实体授予的权限更复杂。有关 Amazon 如何评估策略的更多信息，请参阅策略评估逻辑。

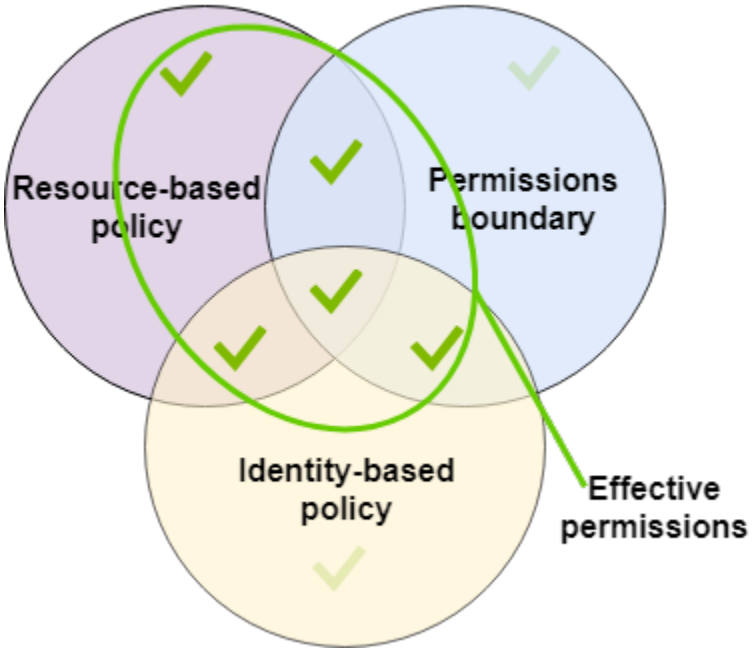
基于身份的策略及边界 - 基于身份的策略是附加到用户、用户组或角色的内联或托管策略。基于身份的策略向实体授予权限，而权限边界限制这些权限。有效的权限是两种策略类型的交集。其中任一项策略中的显式拒绝将覆盖允许。



基于资源的策略 - 基于资源的策略控制指定的主体可以访问策略附加到的资源。

针对 IAM 用户的基于资源的策略

在同一账户中，向 IAM 用户 ARN（即非 Amazon STS 联合用户主体会话）授予权限的基于资源的策略不受基于身份的策略或权限边界中的隐式拒绝限制。



针对 IAM 用户的基于资源的策略

IAM 角色 — 授予 IAM 角色 ARN 权限的基于资源的策略受权限边界或会话策略中隐式拒绝的限制。

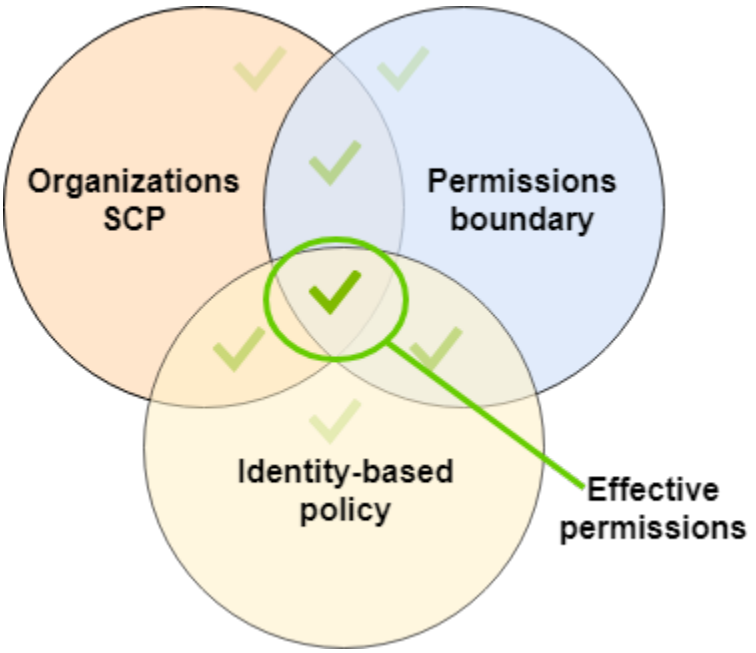
IAM 角色会话 — 在同一账户中，向 IAM 角色会话 ARN 授予权限的基于资源的策略直接向所担任的角色会话授予权限。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。当您担任角色并提出请求时，发出请求的主体是 IAM 角色会话 ARN，而不是角色本身的 ARN。

针对 Amazon STS 联合用户主体会话的基于资源的策略

Amazon STS 联合用户主体会话：Amazon STS 联合用户主体会话是通过调用 `GetFederationToken` 创建的会话。当联合身份用户发出请求时，发出请求的主体是联合身份用户 ARN，而不是联合身份的 IAM 用户的 ARN。在同一个账户中，将权限授予联合身份用户 ARN 的基于资源的策略直接将权限授予会话。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。

但是，如果基于资源的策略向联合身份的 IAM 用户的 ARN 授予权限，则 Amazon STS 联合用户主体在会话期间发出的请求将受权限边界或会话策略中隐式拒绝的限制。

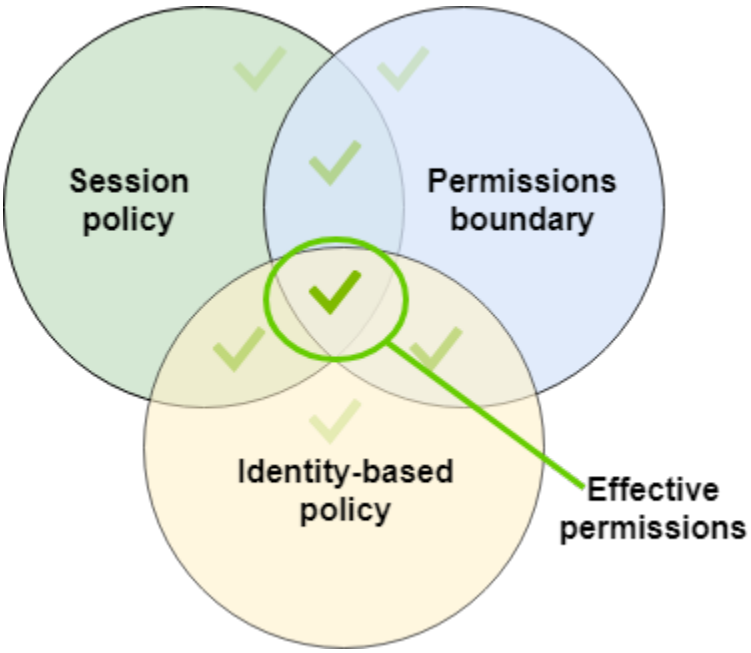
Amazon Organizations SCP：SCP 适用于整个 Amazon Web Services 账户。它们限制账户中的主体所提出的每个请求的权限。IAM 实体（用户或角色）可能会发出受 SCP、权限边界和基于身份的策略影响的请求。在这种情况下，只有在所有三种策略类型都允许时，才允许发出该请求。有效的权限是的所有三种策略类型的交集。任一项策略中的显式拒绝将覆盖允许。



您可以在 Amazon Organizations 中了解您的账户是否为某个组织的成员。组织成员可能会受 SCP 的影响。要使用 Amazon CLI 命令或 Amazon API 操作查看该数据，您必须具有 Amazon Organizations 实体的 `organizations:DescribeOrganization` 操作的权限。您必须具有额外的权限才能在

Amazon Organizations 控制台中执行该操作。要了解 SCP 是否拒绝访问特定的请求或更改您的有效权限，请与您的 Amazon Organizations 管理员联系。

会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。会话的权限来自用于创建会话的 IAM 实体（用户或角色）和会话策略。该实体的基于身份的策略权限受会话策略和权限边界所限制。这组策略类型的有效权限是所有三种策略类型的交集。任一项策略中的显式拒绝将覆盖允许。有关会话策略的更多信息，请参阅会话策略。



使用权限边界将责任委派给其他人

您可以使用权限边界将权限管理任务（如用户创建）委派给您账户中的 IAM 用户。这允许其他人在特定权限边界内代表您执行任务。

例如，假定 María 是 X 公司 Amazon Web Services 账户 的管理员。她希望将用户创建责任委派给 Zhang。但是，她必须确保 Zhang 创建的用户符合以下公司规则：

- 用户不能使用 IAM 来创建或管理用户、组、角色或策略。
- 拒绝用户访问 Amazon S3 `logs` 存储桶，并且用户无法访问 `i-1234567890abcdef0` Amazon EC2 实例。
- 用户无法删除自己的边界策略。

为执行这些规则，María 完成以下任务，下面包括其详细信息：

1. María 创建 `XCompanyBoundaries` 托管策略以用作账户中的所有新用户的权限边界。

2. María 创建 DelegatedUserBoundary 托管策略并将其指定为 Zhang 的权限边界。Maria 记录了其管理员 用户的 ARN，并在策略中使用它来阻止 Zhang 访问它。
3. María 创建 DelegatedUserPermissions 托管策略并将其作为 Zhang 的权限策略进行附加。
4. María 告诉 Zhang 他的新责任和限制。

任务 1：María 必须首先创建托管策略来为新用户定义边界。María 将允许 Zhang 向用户授予他们所需的权限策略，但她希望限制这些用户。为此，她创建以下名为 XCompanyBoundaries 的客户管理型策略。该策略执行以下操作：

- 允许用户完全访问一些服务
- 允许在 IAM 控制台中进行有限的自我管理访问。这意味着，他们可以在登录控制台后更改密码。他们不能设置初始密码。要允许此操作，请将 "*LoginProfile" 操作添加到 AllowManageOwnPasswordAndAccessKeys 语句。
- 拒绝用户访问 Amazon S3 日志存储桶或 i-1234567890abcdef0 Amazon EC2 实例

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ServiceBoundaries",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*",
                "dynamodb:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
}
```

每个语句用于不同目的：

1. 此策略的 `ServiceBoundaries` 语句允许完全访问指定的 Amazon 服务。这意味着，新用户在这些服务中的操作仅受附加到该用户的权限策略的限制。
2. `AllowIAMConsoleForCredentials` 语句允许访问以列出所有 IAM 用户。要在 Amazon Web Services 管理控制台中导航用户页面，此访问权限是必需的。它还允许查看账户的密码要求，这在更改自己的密码时是必需的。

3. AllowManageOwnPasswordAndAccessKeys 语句允许用户仅管理其自己的控制台密码和编程访问密钥。这在 Zhang 或其他管理员为新用户分配了具有完全 IAM 访问权限的权限策略时将非常重要。在这种情况下，该用户可能会更改自己或其他用户的权限。此语句可防止这种情况的发生。
4. DenyS3Logs 语句显式拒绝对 logs 存储桶的访问。
5. DenyEC2Production 语句显式拒绝对 i-1234567890abcdef0 实例的访问。

任务 2：María 希望允许 Zhang 创建所有 X 公司用户，但仅具有 XCompanyBoundaries 权限边界。她创建以下名为 DelegatedUserBoundary 的客户管理型策略。此策略定义 Zhang 可以具有的最大权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "CreateOrChangeOnlyWithBoundary",
            "Effect": "Allow",
            "Action": [
                "iam:AttachUserPolicy",
                "iam:CreateUser",
                "iam:DeleteUserPolicy",
                "iam:DetachUserPolicy",
                "iam:PutUserPermissionsBoundary",
                "iam:PutUserPolicy"
            ],
            "Resource": "*",
            "Condition": {
               "StringEquals": {
                 "iam:PermissionsBoundary": "arn:aws:iam::123456789012:policy/
XCompanyBoundaries"
                }
            }
        },
        {
            "Sid": "CloudWatchAndOtherIAMTasks",
            "Effect": "Allow",
            "Action": [
              "cloudwatch:*",
              "iam:CreateAccessKey",
```

```
            "iam:CreateGroup",
            "iam:CreateLoginProfile",
            "iam:CreatePolicy",
            "iam:DeleteGroup",
            "iam:DeletePolicy",
            "iam:DeletePolicyVersion",
            "iam:DeleteUser",
            "iam:GetAccountPasswordPolicy",
            "iam:GetGroup",
            "iam:GetLoginProfile",
            "iam:GetPolicy",
            "iam:GetPolicyVersion",
            "iam:GetRolePolicy",
            "iam:GetUser",
            "iam:GetUserPolicy",
            "iam:ListAccessKeys",
            "iam:ListAttachedRolePolicies",
            "iam:ListAttachedUserPolicies",
            "iam:ListEntitiesForPolicy",
            "iam:ListGroups",
            "iam:ListGroupsForUser",
            "iam:ListMFADevices",
            "iam:ListPolicies",
            "iam:ListPolicyVersions",
            "iam:ListRolePolicies",
            "iam:ListSSHPublicKeys",
            "iam:ListServiceSpecificCredentials",
            "iam:ListSigningCertificates",
            "iam:ListUserPolicies",
            "iam:ListUsers",
            "iam:SetDefaultPolicyVersion",
            "iam:SimulateCustomPolicy",
            "iam:SimulatePrincipalPolicy",
            "iam:UpdateGroup",
            "iam:UpdateLoginProfile",
            "iam:UpdateUser"
        ],
        "NotResource": "arn:aws:iam::123456789012:user/Maria"
    },
    {
        "Sid": "NoBoundaryPolicyEdit",
        "Effect": "Deny",
        "Action": [
            "iam:CreatePolicyVersion",
```

```
                "iam:DeletePolicy",
                "iam:DeletePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:policy/XCompanyBoundaries",
                "arn:aws:iam::123456789012:policy/DelegatedUserBoundary"
            ]
        },
        {
            "Sid": "NoBoundaryUserDelete",
            "Effect": "Deny",
            "Action": "iam:DeleteUserPermissionsBoundary",
            "Resource": "*"
        }
    ]
}
```

每个语句用于不同目的：

1. CreateOrChangeOnlyWithBoundary 语句允许 Zhang 创建 IAM 用户，但仅当他使用 XCompanyBoundaries 策略设置权限边界时。此语句还允许他为现有用户设置权限边界，但仅使用该相同策略。最后，此语句允许 Zhang 管理设置了此权限边界的用户的权限策略。
2. CloudWatchAndOtherIAMTasks 语句允许 Zhang 完成其他用户、组和策略管理任务。他有权重置密码并为 NotResource 策略元素中未列出的任何 IAM 用户创建访问密钥。这使他能够帮助用户解决登录问题。
3. NoBoundaryPolicyEdit 语句拒绝 Zhang 访问以更新 XCompanyBoundaries 策略。不允许他更改用于为自己或其他用户设置权限边界的任何策略。
4. NoBoundaryUserDelete 语句拒绝 Zhang 访问以便为他或其他用户删除权限边界。

María 然后将 DelegatedUserBoundary 策略指定为 Zhang 用户的权限边界。

任务 3：由于权限边界限制最大权限，但自己不授予访问权限，Maria 必须为 Zhang 创建权限策略。她创建以下名为 DelegatedUserPermissions 的策略。此策略定义 Zhang 在定义的边界内可以执行的操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "IAM",
            "Effect": "Allow",
            "Action": "iam:*",
            "Resource": "*"
        },
        {
            "Sid": "CloudWatchLimited",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:GetDashboard",
                "cloudwatch:GetMetricData",
                "cloudwatch:ListDashboards",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:ListMetrics"
            ],
            "Resource": "*"
        },
        {
            "Sid": "S3BucketContents",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::ZhangBucket"
        }
    ]
}
```

每个语句用于不同目的：

1. 此策略的 IAM 语句允许 Zhang 完全访问 IAM。但是，由于他的权限边界只允许某些 IAM 操作，他的有效 IAM 权限仅受他的权限边界限制。
2. CloudWatchLimited 语句允许 Zhang 在 CloudWatch 中执行五种操作。他的权限边界允许 CloudWatch 中的所有操作，因此他的有效 CloudWatch 权限仅受他的权限策略限制。
3. S3BucketContents 语句允许 Zhang 列出 ZhangBucket Amazon S3 存储桶。但是，他的权限边界不允许任何 Amazon S3 操作，因此无论他的权限策略如何，他都无法执行任何 S3 操作。

> **Note**
>
> Zhang 的策略允许他创建一个用户，然后可以访问自己无法访问的 Amazon S3 资源。通过委托这些管理操作，Maria 实际上相信张能够访问 Amazon S3。

María 然后将 `DelegatedUserPermissions` 策略作为 Zhang 用户的权限策略进行附加。

任务 4：她向 Zhang 提供创建新用户的说明。她告诉他，他可以创建具有所需任何权限的新用户，但他必须为他们分配 `XCompanyBoundaries` 策略作为权限边界。

Zhang 完成以下任务：

1. Zhang 使用 Amazon Web Services 管理控制台 创建用户。他为该用户键入用户名 `Nikhil` 并启用控制台访问。他清除 Requires password reset（需要重置密码）旁边的复选框，因为上述策略仅允许用户在登录到 IAM 控制台后才能更改密码。
2. 在 Set permissions (设置权限) 页面上，Zhang 选择允许 Nikhil 执行其工作的 IAMFullAccess 和 AmazonS3ReadOnlyAccess 权限策略。
3. Zhang 跳过设置权限边界部分，忘记 María 的说明。
4. Zhang 查看用户详细信息并选择创建用户。

   操作失败，并且拒绝访问。Zhang 的 `DelegatedUserBoundary` 权限边界要求他创建的任何用户都将 `XCompanyBoundaries` 策略用作权限边界。
5. Zhang 返回到上一页。在设置权限边界部分中，他选择了 `XCompanyBoundaries` 策略。
6. Zhang 查看用户详细信息并选择创建用户。

   将创建用户。

当 Nikhil 登录时，他有权访问 IAM 和 Amazon S3，但权限边界拒绝的操作除外。例如，他可以在 IAM 中更改自己的密码，但无法创建另一个用户或编辑他的策略。Nikhil 具有对 Amazon S3 的只读访问权限。

如果某个人在 `logs` 存储桶中添加基于资源的策略以允许 Nikhil 将对象放入该存储桶中，他仍然无法访问该存储桶。原因是，其权限边界显式拒绝对 `logs` 存储桶执行的任何操作。任何策略类型中的显式拒绝都会导致请求被拒绝。不过，如果附加到 Secrets Manager 密钥的基于资源的策略允许 Nikhil 执行 `secretsmanager:GetSecretValue` 操作，则 Nikhil 可以检索和解密该密钥。原因是，其权限边界未显式拒绝 Secrets Manager 操作，并且权限边界中的隐式拒绝不限制基于资源的策略。

基于身份的策略和基于资源的策略

策略是Amazon中的对象；在与标识或资源相关联时，策略定义它们的权限。在创建权限策略以限制对资源的访问时，您可以选择基于身份的策略或基于资源的策略。

基于身份的策略附加到 IAM 用户、组或角色。这些策略可让您指定该身份可执行哪些操作（其权限）。例如，您可以将策略附加到名为 John 的 IAM 用户，以声明允许他执行 Amazon EC2 `RunInstances` 操作。该策略可能会进一步声明，John 可以从名为 `MyCompany` 的 Amazon DynamoDB 表中获取项目。您也可以允许 John 管理自己的 IAM 安全凭证。基于标识的策略可以是托管或内联的。

基于资源的策略附加到某个资源。例如，您可以将基于资源的策略附加到 Amazon S3 存储桶、Amazon SQS 队列、VPC 端点、Amazon Key Management Service 加密密钥以及 Amazon DynamoDB 表和流。有关支持基于资源的策略的服务列表，请参阅使用 IAM 的 Amazon 服务。

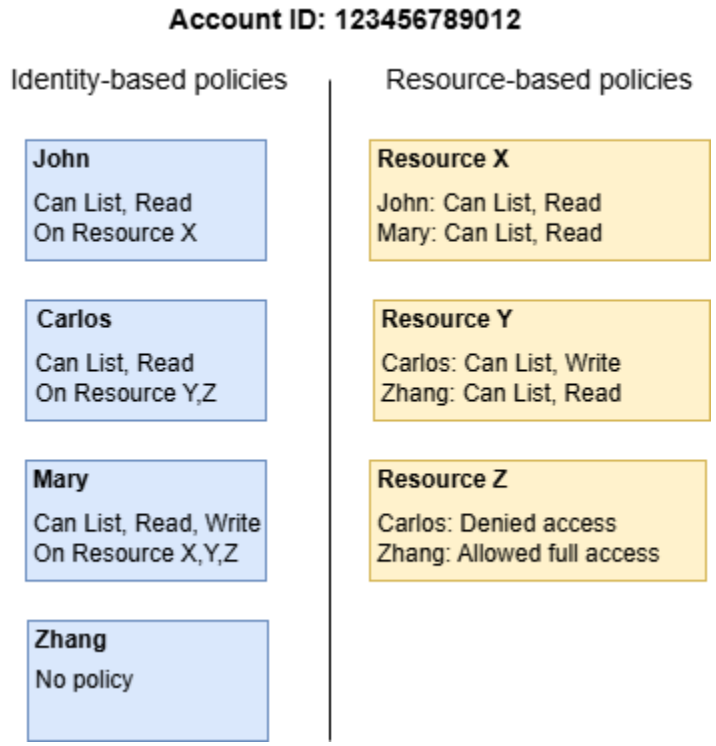
通过使用基于资源的策略，您可以指定哪些用户有权访问资源，以及他们可以对资源执行哪些操作。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。基于资源的策略只有内联的，没有托管的。

> **Note**
>
> 基于资源 的策略与资源级 权限不同。如本主题所述，您可以直接将基于资源的策略附加到某一资源。资源级权限是指使用 ARN 在策略中指定各项资源的能力。只有部分 Amazon 服务支持基于资源的策略。有关支持基于资源的策略和资源级权限的服务列表，请参阅使用 IAM 的 Amazon 服务。

要了解基于身份的策略和基于资源的策略如何在同一个账户中进行交互，请参阅 对单个账户内的请求进行策略评估。

要了解这些策略如何在账户之间进行交互，请参阅 跨账户策略评估逻辑。

要更好地了解这些概念，请查看下图。`123456789012` 账户的管理员已将基于身份的策略 附加到 `John`、`Carlos` 和 `Mary` 用户。可对特定资源执行这些策略中的一些操作。例如，用户 `John` 可以对 `Resource X` 执行一些操作。这是基于身份的策略中的资源级权限。管理员还将基于资源的策略 添加到 `Resource X`、`Resource Y` 和 `Resource Z`。利用基于资源的策略，您可以指定谁可以访问资源。例如，`Resource X` 上的基于资源的策略允许用户 `John` 和 `Mary` 列出和读取资源。



123456789012 账户示例允许以下用户执行列出的操作：

- John – John 可以对 Resource X 执行列出和读取操作。已通过基于身份的策略（针对其用户）和基于资源的策略（针对 Resource X）向他授予此权限。
- Carlos – Carlos 可以对 Resource Y 执行列出、读取和写入操作，但被拒绝访问 Resource Z。Carlos 的基于身份的策略允许他对 Resource Y 执行列出和读取操作。基于 Resource Y 资源的策略还向他授予写入权限。不过，尽管其基于身份的策略允许他访问 Resource Z，但基于 Resource Z 资源的策略拒绝该访问。显式 Deny 会覆盖 Allow，并拒绝他对 Resource Z 进行的访问。有关更多信息，请参阅 策略评估逻辑。
- Mary – Mary 可以对 Resource X、Resource Y 和 Resource Z 执行列出、读取和写入操作。与基于资源的策略相比，她的基于身份的策略允许她对更多资源执行更多操作，但都不拒绝访问。
- Zhang – Zhang 具有 Resource Z 的完全访问权限。Zhang 没有基于身份的策略，但基于 Resource Z 资源的策略向他授予对资源的完全访问权限。Zhang 还可以对 Resource Y 执行列出和读取操作。

基于身份的策略和基于资源的策略都属于权限策略，并且将一起评估。对于仅将权限策略应用于的请求，Amazon 将先检查 Deny 的所有策略。如果存在一个此类策略，则该请求将被拒绝。然后，Amazon 检查每个 Allow。如果至少一个策略语句允许请求中的操作，则将允许此请求。无论 Allow 是在基于身份的策略中还是基于资源的策略中，都无关紧要。

> **Important**
>
> 此逻辑仅当请求在单个 Amazon Web Services 账户 中发出时适用。对于从一个账户向另一个账户发出的请求，Account A 中的请求者必须具有基于身份的策略，该策略允许其向 Account B 中的资源发出请求。此外，Account B 中的基于资源的策略必须允许 Account A 中的请求者访问资源。这两个账户中都必须有允许操作的策略，否则请求失败。有关将基于资源的策略用于跨账户访问的更多信息，请参阅 IAM 中的跨账户资源访问。

具有特定权限的用户可能会请求也附加了权限策略的资源。在这种情况下，在确定是否授予资源的访问权限时，Amazon 将评估这两组权限。有关如何评估策略的信息，请参阅策略评估逻辑。

> **Note**
>
> Amazon S3 支持基于身份的策略和基于资源的策略（称为存储桶策略）。此外，Amazon S3 还支持称为访问控制列表 (ACL) 的权限机制，该机制独立于 IAM 策略和权限。您可以将 IAM 策略与 Amazon S3 ACL 结合使用。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的访问控制。

使用策略控制对 Amazon 资源的访问

您可以使用策略控制对 IAM 或所有 Amazon 中的资源的访问。

要使用标签在 Amazon 中控制访问，您必须了解 Amazon 如何授予访问权限。Amazon 由资源集合组成。IAM 用户是一种资源。Amazon S3 存储桶是一种资源。在使用 Amazon API、Amazon CLI 或 Amazon Web Services 管理控制台 执行操作（如创建用户）时，您将为该操作发送一个请求。该请求会指定操作、资源、主体实体（用户或角色）、主体账户以及所需的任何请求信息。所有这些信息提供了上下文。

然后，Amazon 会检查是否对您（主体）进行身份验证（登录）和授权（具有权限），以便对指定的资源执行指定的操作。在授权期间，Amazon 检查应用于请求上下文的所有策略。大多数策略作为 JSON 文档存储在 Amazon 中，并指定主体实体的权限。有关策略类型和用法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

只有在策略允许请求的每个部分时，Amazon 才会授权该请求。要查看该过程的图表，请参阅IAM 的工作原理。有关 Amazon 如何确定是否允许请求的详细信息，请参阅策略评估逻辑。

在创建 IAM policy 时，您可以控制对以下内容的访问：

- 主体 - 控制允许发出请求的人员（主体）执行哪些操作。
- IAM 身份 – 控制可访问哪些 IAM 身份（IAM 组、用户和角色）以及如何进行访问。
- IAM policy - 控制哪些用户可以创建、编辑和删除客户管理型策略，以及哪些用户可以附加和分离所有托管策略。
- Amazon 资源 - 控制哪些用户有权使用基于身份的策略或基于资源的策略访问资源。
- Amazon 账户 - 控制是否仅允许特定账户的成员发出请求。

通过使用策略，您可以指定哪些用户有权访问 Amazon 资源，以及他们可以对这些资源执行哪些操作。每个 IAM 用户在一开始都没有许可。换而言之，在默认状态下，用户什么都不能做，甚至不能查看自己的访问密钥。要为用户授予执行某些操作的权限，您可以为用户添加权限 (即，将策略附加到用户)。或者，您可以将用户添加到具有所需权限的用户组中。

例如，您可能授予用户列出自己的访问密钥的许可。您可能扩展该许可，并让每位用户创建、更新和删除自己的密钥。

当您授予一个用户组许可时，用户组内的全部用户都会获得那些许可。例如，您可以向 Administrators 用户组授予对任意 Amazon Web Services 账户 资源执行任意 IAM 操作的权限。另一个示例：您可以授予 Managers 用户组描述 Amazon Web Services 账户 的 Amazon EC2 实例的权限。

有关如何向用户、IAM 组和角色委派基本权限的信息，请参阅访问 IAM 资源所需的权限。有关说明基本权限的更多策略示例，请参阅管理 IAM 资源的策略示例。

控制 主体进行的访问

您可以使用策略控制允许发出请求的人员（主体）执行哪些操作。为此，您必须将基于身份的策略附加到此人的身份（用户、用户组或角色）。您还可以使用权限边界以设置实体（用户或角色）可以拥有的最大权限。

例如，假定您希望用户 Zhang Wei 可以完全访问 CloudWatch、Amazon DynamoDB、Amazon EC2 和 Amazon S3。您可以创建两个不同的策略，以便以后其他用户需要一组权限时为其分配其中的一个策略。或者，您可以将这两个权限放在单个策略中，然后将该策略附加到名为 Zhang Wei 的 IAM 用户。您还可以将一个策略附加到 Zhang 所属的用户组，或者附加到 Zhang 可以担任的角色。因此，在 Zhang 查看 S3 存储桶内容时，将允许他的请求。如果他尝试创建新的 IAM 用户，则会拒绝他的请求，因为他没有权限。

您可以对 Zhang 使用权限边界，以确保他无法访问 `amzn-s3-demo-bucket1` S3 存储桶。为此，请确定您希望 Zhang 具有的最大权限。在这种情况下，您使用权限策略控制他执行的操作。在这里，您

只希望他无法访问机密存储桶。因此，您使用以下策略来定义 Zhang 的边界，以允许对 Amazon S3 和一些其他服务执行所有 Amazon 操作，但拒绝访问 amzn-s3-demo-bucket1 S3 存储桶。由于权限边界不允许任何 IAM 操作，它可防止 Zhang 删除他的（或任何人的）边界。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PermissionsBoundarySomeServices",
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "dynamodb:*",
                "ec2:*",
                "s3:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "PermissionsBoundaryNoConfidentialBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket1",
                "arn:aws:s3:::amzn-s3-demo-bucket1/*"
            ]
        }
    ]
}
```

当您分配此类策略作为用户的权限边界时，请记住，它不授予任何权限。它设置基于身份的策略可以授予 IAM 实体的最大权限。有关权限边界的更多信息，请参阅IAM 实体的权限边界。

有关上述过程的详细信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。
- 要了解如何将 IAM policy 附加到主体，请参阅 添加和删除 IAM 身份权限。

- 要查看授予 EC2 完全访问权限的示例策略，请参阅Amazon EC2：允许以编程方式和使用控制台在特定区域进行不受限制的 EC2 访问。
- 要允许对 S3 存储桶进行只读访问，请使用以下示例策略的前两个语句：Amazon S3：允许以编程方式和在控制台中对 S3 存储桶中的对象进行读写访问。
- 要查看允许用户设置凭证（如控制台密码、编程访问密钥和 MFA 设备）的示例策略，请参阅 Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。

控制对身份的访问

您可以通过用户组创建附加到所有用户的 IAM policy，以使用该策略控制您的用户可对某个身份执行哪些操作。为此，请创建一个策略以限制可对某个身份执行哪些操作，以及哪些用户可以访问该身份。

例如，您可以创建一个名为 AllUsers 的组，然后将该用户组附加到所有用户。在创建该用户组时，您可以如前一节所述，为所有用户授予访问权限以设置其凭证。然后，您可以创建一个策略，以便拒绝访问以禁止更改该用户组，除非在策略条件中包含用户名。但是，该策略部分仅拒绝列出的用户以外的任何人进行访问。您还必须包含相应的权限，以允许该用户组中的所有用户执行所有用户组管理操作。最后，您将该策略附加到该用户组，以便将其应用于所有用户。因此，在策略中未指定的用户尝试更改该用户组时，将拒绝请求。

使用可视化编辑器创建该策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。

   如果这是您首次选择策略，则会显示欢迎访问托管式策略页面。选择开始使用。
3. 选择创建策略。
4. 在策略编辑器部分，选择可视化选项。
5. 在选择服务中，选择 IAM。
6. 在允许的操作中，在搜索框中键入 **group**。可视化编辑器将显示包含 group 一词的所有 IAM 操作。选中所有复选框。
7. 选择资源以指定您的策略的资源。根据选择的操作，您应该会看到组、和用户资源类型。

   - 组 - 选择添加 ARN。对于资源位置，选择任何账户选项。选中任何具有路径的组名称复选框，然后键入用户组名称 **AllUsers**。然后选择添加 ARN。
   - 用户 – 选中此账户中的任何项旁边的复选框。

您选择的操作之一 (`ListGroups`) 不支持使用特定的资源。您不需要为该操作选择所有资源。在 JSON 编辑器中保存或查看您的策略时，您可以看到 IAM 自动创建新的权限块，以便为所有资源授予该操作权限。

8. 要添加其他权限块，请选择添加更多权限。
9. 选择选择服务，然后选择 IAM。
10. 选择允许的操作，然后选择切换到拒绝权限。在执行该操作时，将使用整个块拒绝权限。
11. 在搜索框中，键入 **`group`**。可视化编辑器将显示包含 `group` 一词的所有 IAM 操作。选中以下操作旁边的复选框：

    - CreateGroup
    - DeleteGroup
    - RemoveUserFromGroup
    - AttachGroupPolicy
    - DeleteGroupPolicy
    - DetachGroupPolicy
    - PutGroupPolicy
    - UpdateGroup
12. 选择资源以指定您的策略的资源。根据选择的操作，将会看到 group 资源类型。选择添加 ARN。对于资源位置，选择任何账户选项。对于任何具有路径的组名称，键入用户组名称 **`AllUsers`**。然后选择添加 ARN。
13. 选择请求条件 – 可选，然后选择添加其他条件。使用以下值填写表单：

    - 条件密钥 – 选择 aws:username
    - Qualifier（限定符）- 选择 Default（默认值）
    - Operator（运算符）- 选择 StringNotEquals
    - 值 – 键入 **`srodriguez`**，然后选择 添加以添加另一个值。键入 **`mjackson`**，然后选择添加以添加另一个值。键入 **`adesai`**，然后选择添加条件。

    在列表中不包含进行调用的用户时，该条件确保拒绝访问指定的用户组管理操作。由于这会显式拒绝权限，它将覆盖前面允许这些用户调用这些操作的块。不会拒绝列表中的用户进行访问，并在第一个权限块中为他们授予权限，因此，他们可以完全管理该用户组。
14. 完成后，选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

15. 在查看并创建页面上，对于策略名称，键入 **`LimitAllUserGroupManagement`**。对于描述，请键入 **`Allows all users read-only access to a specific user group, and allows only specific users access to make changes to the user group`**。查看此策略中定义的权限，确保您授予了所需的权限。然后，选择创建策略以保存新策略。
16. 将策略附加到您的用户组。有关更多信息，请参阅 添加和删除 IAM 身份权限。

或者，您也可以使用该示例 JSON 策略文档创建相同的策略。要查看该 JSON 策略，请参阅 IAM：允许特定的 IAM 用户以编程方式和在控制台中管理组。有关使用 JSON 文档创建策略的详细说明，请参阅the section called “使用 JSON 编辑器创建策略”。

控制对策略的访问

您可以控制您的用户如何应用 Amazon 托管策略。为此，请将该策略附加到您的所有用户。理想情况下，您可以使用用户组执行该操作。

例如，您可能会创建一个策略允许用户仅将 IAMUserChangePassword 和 PowerUserAccess Amazon 托管策略附加到新的 IAM 用户、用户组或角色。

对于客户托管策略，您可以控制哪些用户可以创建、更新和删除这些策略。您可以控制哪些用户可以将策略附加到委托人实体（IAM 组、用户和角色）以及将策略从主体实体中分离。还可以控制用户可以对哪些实体附加或分离哪些策略。

例如，您可以为账户管理员授予权限以创建、更新和删除策略。然后，您可以为小组负责人或其他有限管理员授予权限，以便将这些策略附加到该有限管理员管理的主体实体以及将这些策略从这些主体实体中分离。

有关更多信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。
- 要了解如何将 IAM policy 附加到主体，请参阅 添加和删除 IAM 身份权限。

- 要查看限制使用托管策略的示例策略，请参阅IAM：对可应用于 IAM 用户、组或角色的托管策略加以限制。

控制创建、更新和删除客户托管策略的权限

您可以使用 IAM policy 控制谁有权在您的 Amazon Web Services 账户 中创建、更新和删除客户管理型策略。以下列表包含与创建、更新和删除策略或策略版本直接相关的 API 操作：

- CreatePolicy
- CreatePolicyVersion
- DeletePolicy
- DeletePolicyVersion
- SetDefaultPolicyVersion

上述列表中的 API 操作对应于可使用 IAM policy 允许或拒绝的操作，即，您可以授予的权限。

考虑以下示例策略。它允许用户在 Amazon Web Services 账户 中创建、更新（即创建新的策略版本）、删除所有客户管理型策略以及设置这些策略的默认版本。该示例策略还允许该用户列出策略并获取策略。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

Example允许创建、更新、删除、列出、获取所有策略以及设置这些策略的默认版本的示例策略

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": [
      "iam:CreatePolicy",
      "iam:CreatePolicyVersion",
      "iam:DeletePolicy",
      "iam:DeletePolicyVersion",
      "iam:GetPolicy",
      "iam:GetPolicyVersion",
      "iam:ListPolicies",
      "iam:ListPolicyVersions",
```

```
      "iam:SetDefaultPolicyVersion"
    ],
    "Resource": "*"
  }
}
```

您可以创建限制使用这些 API 操作的策略以仅影响指定的托管策略。例如，您可能希望允许用户设置默认版本和删除策略版本，但是仅允许针对特定客户托管策略执行这些操作。通过在授予这些权限的策略的 Resource 元素中指定策略 ARN，可以实现此目的。

下例所示的策略允许用户删除策略版本和设置默认版本。但是，仅包含路径 /TEAM-A/ 的客户托管策略允许这些操作。客户托管策略 ARN 在策略的 Resource 元素中指定。（在此示例中，ARN 包含一个路径和一个通配符，因而与包含路径 /TEAM-A/ 的所有客户托管策略匹配）。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

有关在客户托管策略名称中使用路径的更多信息，请参阅易记名称和路径。

Example仅允许针对特定策略删除策略版本和设置默认版本的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:DeletePolicyVersion",
            "iam:SetDefaultPolicyVersion"
        ],
        "Resource": "arn:aws:iam::111122223333:policy/TEAM-A/*"
    }
}
```

控制附加和分离托管策略的权限

您还可以使用 IAM policy 仅允许用户使用特定的托管策略。实际上，就是可以控制允许用户向其他主体实体授予的权限。

以下列表显示与将托管策略附加到主体实体以及从中分离直接相关的 API 操作：

- AttachGroupPolicy
- AttachRolePolicy
- AttachUserPolicy
- DetachGroupPolicy
- DetachRolePolicy
- DetachUserPolicy

您可以创建限制使用这些 API 操作的策略以仅影响指定的特定托管策略和/或主体实体。例如，您可能希望允许用户附加托管策略，但是只能附加您指定的托管策略。或者，您可能希望允许用户附加托管策略，但是只能附加到您指定的主体实体。

以下示例策略仅允许用户将托管策略附加到包含路径 /TEAM-A/ 的 IAM 组和角色。用户组和角色 ARN 是在策略的 `Resource` 元素中指定的。（在此示例中，ARN 包含一个路径和一个通配符，因而与包含路径 /TEAM-A/ 的所有 IAM 组和角色匹配）。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

Example仅允许将托管策略附加到特定用户组或角色的示例策略

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:AttachGroupPolicy",
            "iam:AttachRolePolicy"
        ],
        "Resource": [
            "arn:aws:iam::111122223333:group/TEAM-A/*",
            "arn:aws:iam::111122223333:role/TEAM-A/*"
        ]
    }
}
```

您可以进一步限制前面示例中的操作，使之仅影响特定的策略。即，通过向策略添加条件，可以控制允许用户附加到其他主体实体的权限。

以下示例中的条件可确保仅当附加的策略与指定策略之一匹配时才允许 AttachGroupPolicy 和 AttachRolePolicy 权限。该条件使用 iam:PolicyARN 条件键确定允许附加的策略。以下示例策略扩展前一示例。该示例仅允许用户将包含路径 /TEAM-A/ 的托管策略附加到包含路径 /TEAM-A/ 的 IAM 组和角色。要了解如何使用该示例 JSON 策略文档创建策略，请参阅 the section called “使用 JSON 编辑器创建策略”。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:AttachGroupPolicy",
            "iam:AttachRolePolicy"
        ],
        "Resource": [
            "arn:aws:iam::111122223333:group/TEAM-A/*",
            "arn:aws:iam::111122223333:role/TEAM-A/*"
        ],
        "Condition": {
            "ArnLike": {
                "iam:PolicyARN": "arn:aws:iam::111122223333:policy/TEAM-A/*"
            }
        }
    }
}
```

此策略使用 ArnLike 条件运算符，但也可以使用 ArnEquals 条件运算符，因为这两个条件运算符的行为相同。有关 ArnLike 和 ArnEquals 的更多信息，请参阅Amazon Resource Name (ARN) 条件运算符策略元素参考 的条件类型 部分中的。

例如，您可以限制使用操作以仅涉及指定的托管策略。通过在授予这些权限的策略的 Condition 元素中指定策略 ARN，可以实现此目的。例如，指定客户托管策略的 ARN：

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::123456789012:policy/POLICY-NAME"}
}
```

您还可以在策略的 Condition 元素中指定 Amazon; 托管策略的 ARN。Amazon 托管策略的 ARN 在策略 ARN 中使用特殊别名 aws 而不是账户 ID，如此示例所示：

```
"Condition": {"ArnEquals":
  {"iam:PolicyARN": "arn:aws:iam::aws:policy/AmazonEC2FullAccess"}
}
```

控制对资源的访问

您可以使用基于身份的策略或基于资源的策略控制对资源的访问。在基于身份的策略中，您将策略附加到一个身份并指定该身份可以访问哪些资源。在基于资源的策略中，您将策略附加到要控制的资源。在该策略中，您指定哪些主体可以访问该资源。有关这两种类型的策略的更多信息，请参阅基于身份的策略和基于资源的策略。

有关更多信息，请参阅以下资源：

- 要了解创建可附加到主体的 IAM policy 的更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。
- 要了解如何将 IAM policy 附加到主体，请参阅 添加和删除 IAM 身份权限。
- Amazon S3 支持在存储桶上使用基于资源的策略。有关更多信息，请参阅存储桶策略示例。

资源创建者不会自动拥有许可

如果使用 Amazon Web Services 账户根用户凭证登录，您有权对属于该账户的资源执行任何操作。但是，对 IAM 用户来说并非如此。IAM 用户可以获得创建资源的权限，但即使对于该资源，该用户权限也仅限于明确授予的内容。这意味着，您不会仅仅因为创建资源 (如 IAM 角色) 而自动具有编辑或删除该角色的权限。此外，账户所有者或有权管理您的权限的其他用户可以随时撤消您的权限。

控制对特定账户中的主体的访问

您可以直接向自己账户中的 IAM 用户授予对您的资源的访问权限。如果其他账户中的用户需要访问您的资源，您可以创建 IAM 角色。角色是包含权限但不与特定用户关联的实体。然后，其他账户中的用户可以担任该角色，并根据您为该角色分配的权限访问资源。有关更多信息，请参阅 在您拥有的其他 Amazon Web Services 账户 中 IAM 用户的访问权限。

> **Note**
>
> 有些服务支持基于资源的策略，如 基于身份的策略和基于资源的策略 中所述（例如 Amazon S3、Amazon SNS 和 Amazon SQS）。对于这些服务，使用角色的替代方法是将策略附加

到要共享的资源（存储桶、主题或队列）。基于资源的策略可以指定具有访问资源权限的 Amazon 账户。

使用标签控制对 IAM 用户和角色的访问以及他们进行的访问

可以使用以下部分中的信息控制谁可以访问您的 IAM 用户和角色以及您的用户和角色可以访问哪些资源。有关控制对其他 Amazon 资源（包括其他 IAM 资源）的访问的更多一般信息和示例，请参阅 Amazon Identity and Access Management 资源的标签。

> **Note**
>
> 有关标签键和标签键值区分大小写的详细信息，请参阅 Case sensitivity。

可以将标签附加到 IAM 资源，在请求中传递标签，或者将标签附加到发出请求的主体。IAM 用户或角色可以同时是资源和主体。例如，您可以编写一个策略以允许用户列出用户的组。只有在发出请求的用户（主体）具有与他们尝试查看的用户相同的 `project=blue` 标签时，才允许执行该操作。在该示例中，用户可以查看任何用户（包括他们自己）的组成员资格，只要他们处理的是同一个项目。

要基于标签控制访问，您需要在策略的条件元素中提供标签信息。在创建 IAM policy 时，您可以使用 IAM 标签和关联的标签条件键控制对任何以下内容的访问：

- Resource（资源）- 根据标签控制对用户或角色资源的访问。为此，请使用 aws:ResourceTag/***key-name*** 条件键指定必须将哪个标签键值对附加到资源。有关更多信息，请参阅 控制对 Amazon 资源的访问。
- 请求 – 控制可以在 IAM 请求中传递什么标签。为此，请使用 aws:RequestTag/***key-name*** 条件键来指定可在 IAM 用户或角色中添加、更改或删除的标签。对于 IAM 资源和其他 Amazon 资源，将按相同的方式使用该键。有关更多信息，请参阅 在 Amazon 请求期间控制访问。
- 主体 - 根据附加到发出请求的人员（主体）的 IAM 用户或角色的标签，控制允许该人员执行哪些操作。为此，请使用 aws:PrincipalTag/***key-name*** 条件键指定在允许请求之前必须将哪些标签附加到 IAM 用户或角色。
- 授权过程的任何部分 – 使用 aws:TagKeys 条件键来控制是否可以在请求中或通过主体使用特定的标签键。在这种情况下，键值无关紧要。对于 IAM 和其他 Amazon 服务，该键的行为是类似的。不过，在 IAM 中标记用户时，这还会控制主体是否可以向任何服务发出请求。有关更多信息，请参阅 根据标签键控制访问。

您可以使用可视化编辑器或 JSON 创建 IAM policy，也可以导入现有的托管策略以创建该策略。有关更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。

> **Note**
>
> 您也可以在代入 IAM 角色或联合用户身份时传递会话标签。这仅对会话的长度有效。

控制 IAM 主体进行的访问

您可以根据附加到主体身份的标签来控制允许该人员执行哪些操作。

此示例显示了您可以如何创建基于身份的策略，以允许此账户中的任何用户查看任何用户（包括他们自己）的组成员资格，只要他们处理的是同一个项目。只有在用户的资源标签和主体的标签具有相同的标签键 `project` 值时，才允许执行该操作。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": "iam:ListGroupsForUser",
            "Resource": "arn:aws:iam::111222333444:user/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/project":
 "${aws:PrincipalTag/project}"}
            }
        }]
}
```

根据标签键控制访问

您可以在 IAM policy 中使用标签来控制是否可以在请求中或通过主体使用特定的标签键。

此示例说明如何创建基于身份的策略，以允许仅从用户删除带有 `temporary` 密钥的标签。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:UntagUser",
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys":
 ["temporary"]}}
    }]
}
```

使用标签控制对 Amazon 资源的访问

您可以使用标签以控制对支持标记的 Amazon 资源（包括 IAM 资源）的访问。您可以标记 IAM 用户和角色以控制他们可以访问哪些内容。要了解如何标记 IAM 用户和角色，请参阅 Amazon Identity and Access Management 资源的标签。此外，您还可以控制对以下 IAM 资源的访问：客户托管策略、IAM 身份提供程序、实例配置文件、服务器证书和虚拟 MFA 设备。对于创建策略，以允许具有主体标签的 IAM 角色访问具有匹配标签的资源并测试该策略，有关教程，请参阅 IAM 教程：根据标签定义访问 Amazon 资源的权限。可以使用以下部分中的信息以控制对其他 Amazon 资源（包括 IAM 资源）的访问，而无需标记 IAM 用户或角色。

在使用标签控制对 Amazon 资源的访问之前，您必须了解 Amazon 如何授予访问权限。Amazon 由一组资源 组成。Amazon EC2 实例是一种资源。Amazon S3 存储桶是一种资源。您可以使用 Amazon API、Amazon CLI 或 Amazon Web Services 管理控制台 执行操作，例如，在 Amazon S3 中创建存储桶。在执行操作时，您将发送该操作的请求。您的请求指定操作、资源、主体实体（用户或角色)、主体账户 以及所需的任何请求信息。所有这些信息提供了上下文。

Amazon然后， 检查是否对您（主体实体）进行身份验证（登录）和授权（具有权限），以便对指定的资源执行指定的操作。在授权期间，Amazon 检查应用于请求上下文的所有策略。大多数策略作为 JSON 文档存储在 Amazon 中，并指定主体实体的权限。有关策略类型和用法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。

只有在策略允许请求的每个部分时，Amazon 才会授权该请求。要查看示意图和详细了解 IAM 基础设施，请参阅 IAM 的工作原理。有关 IAM 如何确定是否允许请求的详细信息，请参阅 策略评估逻辑。

标签是此过程中的另一个考虑事项，因为标签可以附加到资源，也可以从请求 传入支持标签的服务。要基于标签控制访问，您需要在策略的条件元素中提供标签信息。要了解 Amazon 服务是否支持使用标签控制访问权限，请参阅 使用 IAM 的 Amazon 服务 并查找在 ABAC 列中为是的服务。选择服务名称以查看该服务的授权和访问控制文档。

然后，您可以创建一个 IAM policy，以根据资源的标签允许或拒绝访问资源。在该策略中，您可以使用标签条件键以控制对任何以下内容的访问：

- Resource（资源）- 基于 Amazon 服务资源上的标签控制对这些资源的访问。为此，请使用 aws:ResourceTag/***key-name*** 条件键根据附加到资源的标签确定是否允许访问资源。
- Request（请求）- 控制可以在请求中传递哪些标签。为此，请使用 aws:RequestTag/***key-name*** 条件键，指定可以在请求中传递的标签键值对以用于标记 Amazon 资源。
- 授权过程的任何部分 - 使用 aws:TagKeys 条件键来控制是否可以在请求中使用特定的标签键。

您可以通过可视化方式创建 IAM policy，也可以使用 JSON，或通过导入现有托管策略来创建。有关更多信息，请参阅 使用客户管理型策略定义自定义 IAM 权限。

> **Note**
>
> 如果用户有权使用创建资源的操作，则某些服务允许用户在创建资源时指定标签。

控制对 Amazon 资源的访问

您可以使用 IAM policy 中的条件根据 Amazon 资源上的标签控制对该资源的访问。您可以使用全局 `aws:ResourceTag/`*`tag-key`* 条件键或服务特定的键执行该操作。某些服务仅支持此键的服务特定版本，而不支持全局版本。

> **Warning**
>
> 请勿试图通过标记角色然后在带有 `iam:PassRole` 操作的策略中使用 `ResourceTag` 条件键来控制谁可以传递角色。这种方法没有可靠的结果。有关将角色传递给服务所需的权限的更多信息，请参阅向用户授予权限以将角色传递给 Amazon 服务。

此示例说明如何创建允许启动或停止 Amazon EC2 实例的基于身份的策略。只有在实例标签 Owner 具有该用户的用户名值时，才允许执行这些操作。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Owner": "${aws:username}"}
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

您可以将该策略附加到您账户中的 IAM 用户。如果名为 richard 的用户尝试启动 Amazon EC2 实例，则实例必须标记为 Owner=richard 或 owner=richard。否则，他将被拒绝访问。标签键 Owner 同时匹配 owner 和 Owner，因为条件键名称不区分大小写。有关更多信息，请参阅 IAM JSON 策略元素：Condition。

此示例说明了如何创建基于身份的策略，该策略使用了资源 ARN 中的 team 主体标签。该策略授予删除 Amazon Simple Queue Service 队列的权限，但前提是队列名称以团队名称开头，后跟 -queue。例如，如果 qa 是 team 主体标签的团队名称，则队列名称为 qa-queue。

JSON

```
{
```

```
      "Version":"2012-10-17",
      "Statement": {
        "Sid": "AllQueueActions",
        "Effect": "Allow",
        "Action": "sqs:DeleteQueue",
        "Resource": "arn:aws:sqs:us-east-2:111122223333:${aws:PrincipalTag/team}-
queue"
      }
}
```

在 Amazon 请求期间控制访问

您可以在 IAM 策略中使用条件以控制可在对 Amazon 资源应用标签的请求中传递哪些标签键值对。

此示例说明如何创建允许使用 Amazon EC2 `CreateTags` 操作将标签附加到实例的基于身份的策略。只有在标签包含 `environment` 键和 `preprod` 或 `production` 值时，才能附加标签。如果需要，您可以将 `ForAllValues` 修饰符与 `aws:TagKeys` 条件键一起使用，以指示仅允许在请求中使用 `environment` 键。这将阻止用户包括其他键，例如意外使用 `Environment` 而不是 `environment`。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "ec2:CreateTags",
        "Resource": "arn:aws:ec2:*:*:instance/*",
        "Condition": {
            "StringEquals": {
                "aws:RequestTag/environment": [
                    "preprod",
                    "production"
                ]
            },
            "ForAllValues:StringEquals": {"aws:TagKeys": "environment"}
        }
    }
}
```

根据标签键控制访问

您可以在 IAM policy 中使用条件来控制是否可以在请求中使用特定标签键。

我们建议，当您使用策略来通过标签控制访问时，应该使用 aws:TagKeys 条件键。支持标签的 Amazon 服务可能允许您创建多个只有大小写不同的标签键名称，例如使用 stack=production 和 Stack=test 标记 Amazon EC2 实例。在策略条件中，键名称不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的资源标签键，但不会同时匹配两者。要防止只有键的大小写形式不同的重复标签，请使用 aws:TagKeys 条件来定义用户可以应用的标签键，或使用带有 Amazon Organizations 的标签策略。有关更多信息，请参阅《Amazon Organizations 用户指南》中的标签策略。

此示例说明了如何创建允许创建 Secrets Manager 密钥及将其贴标签的基于身份的策略，但仅限于标签密钥 environment 或 cost-center。Null 条件可确保在请求中没有标签时，条件的计算结果为 false。

```
{
        "Effect": "Allow",
        "Action": [
            "secretsmanager:CreateSecret",
            "secretsmanager:TagResource"
        ],
        "Resource": "*",
        "Condition": {
            "Null": {
                "aws:TagKeys": "false"
            },
            "ForAllValues:StringEquals": {
                "aws:TagKeys": [
                    "environment",
                    "cost-center"
                ]
            }
        }
}
```

IAM 中的跨账户资源访问

对于一些 Amazon 服务，您可以使用 IAM 授予对资源的跨账户访问权限。为此，您可以将资源策略直接附加到要共享的资源，也可以将角色用作代理。

要直接共享资源，要共享的资源必须支持基于资源的策略。与针对角色的基于身份的策略不同，基于资源的策略指定谁（哪个主体）可以访问该资源。

如果要访问另一个账户中不支持基于资源的策略的资源，请使用角色作为代理。

有关这些策略类型之间差异的详细信息，请参阅 基于身份的策略和基于资源的策略。

> **Note**
>
> IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，假定您在标准 aws 分区的美国西部（北加利福尼亚）中有一个账户。您在 aws-cn 分区的中国也有一个账户。您不能使用中国账户中基于资源的策略，来允许标准 Amazon 账户中用户的访问权限。

使用角色进行跨账户访问

并非所有的 Amazon 服务都支持基于资源的策略。对于这些服务，在提供对多个服务的跨账户访问权限时，您可以使用跨账户 IAM 角色来集中管理权限。跨账户 IAM 角色是一种 IAM 角色，包含允许其他 Amazon 账户中的 IAM 主体担任该角色的信任策略。简而言之，您可以在一个 Amazon 账户中创建一个角色，将特定权限委派给另一个 Amazon 账户。

有关将策略附加到 IAM 身份的信息，请参阅 管理 IAM 策略。

> **Note**
>
> 当主体切换为某个角色，临时使用该角色的权限时，他们会放弃其原始权限，并获得分配给他们所担任之角色的权限。

让我们了解一下适用于需要访问客户账户的 APN 合作伙伴软件的整体流程。

1. 客户使用策略创建账户中的 IAM 角色，该角色允许访问 APN 合作伙伴所需的 Amazon S3 资源。在本示例中，角色名称为 APNPartner。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name"
            ]
        }
    ]
}
```

2. 然后，客户通过在 APNPartner 角色的 信任策略 中提供 APN 合作伙伴的 Amazon Web Services 账户 ID，指定合作伙伴的 Amazon 账户可以代入该角色。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/APN-user-name"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

3. 客户将角色的 Amazon 资源名称（ARN）提供给 APN 合作伙伴。ARN 是角色的完全限定名称。

```
arn:aws:iam::Customer-Account-ID:role/APNPartner
```

> **Note**
>
> 我们建议在多租户情况下使用外部 ID。有关更多信息，请参阅 访问第三方拥有的 Amazon Web Services 账户。

4. 当 APN 合作伙伴的软件需要访问客户的账户时，软件会在 Amazon Security Token Service 中使用客户账户中角色的 ARN 调用 AssumeRole API。STS 返回允许软件执行其工作的临时 Amazon 凭证。

有关使用角色授予跨账户访问权限的另一个示例，请参阅 在您拥有的其他 Amazon Web Services 账户中 IAM 用户的访问权限。您也可以关注 IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限。

使用基于资源的策略授予跨账户访问权限

当一个账户使用基于资源的策略通过另一个账户访问资源时，主体仍在可信账户中工作，并且无需放弃其权限来接收角色权限。换言之，主体既能够继续访问可信账户中的资源，又能继续访问信任账户中的资源。这对于向另一个账户中的共享资源复制信息或从中复制信息之类的任务非常有用。

可在基于资源的策略中指定的主体包括账户、IAM 用户、Amazon STS 联合用户主体、SAML 联合主体、OIDC 联合主体、IAM 角色、代入角色会话或 Amazon 服务。有关更多信息，请参阅指定主体。

要了解您信任区域之外的账户（受信任的组织或账户）中的主体是否有权承担您的角色，请参阅识别与外部实体共享的资源。

以下列表包括一些支持基于资源的策略的 Amazon 服务。有关支持向资源而非主体附加权限策略的更多 Amazon 服务的完整列表，请参阅 使用 IAM 的 Amazon 服务 并寻找具有基于资源列中为是的服务。

- Amazon S3 存储桶 — 策略将附加到存储桶，但该策略可控制对存储桶和存储桶中对象的访问。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 Amazon S3 的存储桶策略。在某些情况下，可能最好使用角色对 Amazon S3 进行跨账户访问。有关更多信息，请参阅 Amazon Simple Storage Service 用户指南中的示例演练。
- Amazon Simple Notification Service（Amazon SNS）主题 — 有关更多信息，请转至《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 访问控制示例。
- Amazon Simple Queue Service (Amazon SQS) 队列 — 有关更多信息，请转至 Amazon Simple Queue Service 开发人员指南中的附录：访问策略语言

基于资源的策略，用于委派 Amazon 权限

如果资源向账户中的主体授予权限，则您随后可将这些权限委派给特定的 IAM 身份。身份是您账户中的用户、用户组或角色。可以通过将策略附加到身份来委派权限。您可以授予拥有资源的账户所允许的最多权限。

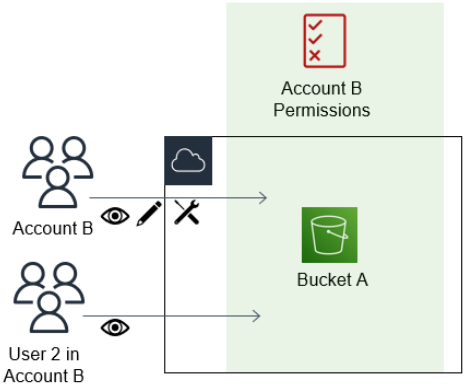
> **Important**
>
> 在跨账户访问中，主体需要在身份策略和基于资源的策略中获得 `Allow`。

假定基于资源的策略允许您账户中的所有主体对资源进行完全管理访问。随后，您可以将完全访问权限、只读访问权限或任何其他部分访问权限委派给 Amazon 账户中的主体。或者，如果基于资源的策略仅允许列表权限，则您只能委派列表访问权限。如果您尝试委派的权限超出您的账户所拥有的权限，您的主体将只有列表访问权限。

有关如何做出这些决定的更多信息，请参阅确定账户内的请求是被允许还是被拒绝。

> **Note**
>
> IAM 角色和基于资源的策略仅在单个分区内跨账户委派访问权限。例如，您不能在标准 aws 分区的账户与 aws-cn 分区中的账户之间添加跨账户访问权限。

例如，假定您管理 AccountA 和 AccountB。在 AccountA 中，您将有名为 BucketA 的 Amazon S3 存储桶。



1. 您将基于资源的策略附加到 BucketA，这将允许 AccountB 中的所有主体对存储桶中的对象进行完全访问。这些主体可以创建、读取或删除该存储桶中的任何对象。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalAccess",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:root"
            },
```

```
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::BucketA/*"
        }
    ]
}
```

AccountA 通过在基于资源的策略中将 AccountB 指定为主体来向 AccountB 提供对 BucketA 的完全访问权限。这样，AccountB 有权对 BucketA 执行任何操作，AccountB 管理员可向其在 AccountB 中的用户委派访问权限。

AccountB 根用户具有授予给该账户的所有权限。因此，根用户具有对 BucketA 的完全访问权限。

2. 在 AccountB 中，将策略附加到名为 User2 的 IAM 用户。该策略允许用户对 BucketA 中的对象进行只读访问。这意味着，User2 可以查看对象，但不能创建、编辑或删除对象。

   JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect" : "Allow",
            "Action" : [
                "s3:Get*",
                "s3:List*" ],
                "Resource" : "arn:aws:s3:::BucketA/*"
        }
    ]
}
```

AccountB 可委派的最大访问级别是授予给账户的访问级别。在此情况下，基于资源的策略授予了对 AccountB 的完全访问权限，但仅向 User2 授予了只读访问权限。

AccountB 管理员未向 User1 授予访问权限。默认情况下，除非明确授予，否则用户没有任何权限，因此 User1 没有对 BucketA 的访问权限。

IAM 将在主体发出请求时对主体的权限进行评估。因此，如果您使用通配符（*）向用户授予对资源的完全访问权限，则主体可以访问您的 Amazon 账户有权访问的任何资源。甚至对于您在创建用户策略后添加或获得其访问权的资源也是如此。

在上述示例中，如果 AccountB 已将允许对所有账户中的所有资源进行完全访问的策略附加到 User2，则 User2 将自动具有 AccountB 有权访问的任何资源的访问权限。这包括 BucketA 访问权限以及由 AccountA 中的基于资源的策略授予的对任何其他资源的访问权限。

有关角色的复杂用法（例如，授予对应用程序和服务的访问权限）的更多信息，请参阅 IAM 角色的常见场景。

> **Important**
>
> 仅向信任实体授予访问权限，并提供最低级别的访问权限。只要可信实体是另一个 Amazon 账户，就可以授予任何 IAM 主体访问您的资源的权限。可信 Amazon 账户可委派的访问权限范围仅限于它所获得的访问权限；它委派的权限不能超出账户本身所获得的访问权限。

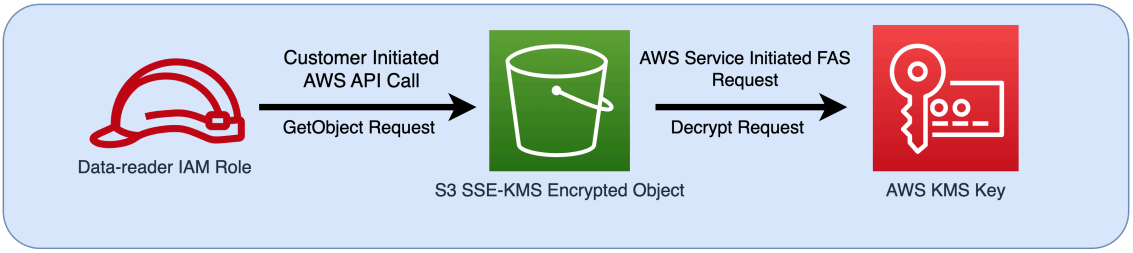
有关权限、策略以及用于编写策略的权限策略语言的信息，请参阅适用于 Amazon 资源的 Access Management。

转发访问会话

转发访问会话（FAS）是一种 IAM 技术，当 Amazon 服务代表您发出请求时，Amazon 服务使用该技术来传递您的身份、权限和会话属性。FAS 使用调用 Amazon 服务的身份的权限以及 Amazon 服务的身份向下游服务发出请求。只有在服务收到需要与其他 Amazon 服务或资源交互才能完成的请求后，才会代表 IAM 主体向 Amazon 服务发出 FAS 请求。当发出 FAS 请求时：

- 接收 IAM 主体初始请求的服务会检查 IAM 主体的权限。
- 接收 FAS 后续请求的服务也会检查同一 IAM 主体的权限。

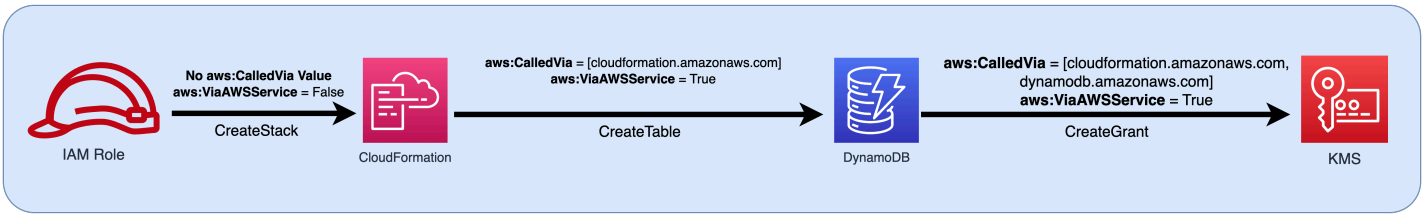
例如，当使用 SSE-KMS 加密对象时，Amazon S3 使用 FAS 调用 Amazon Key Management Service 以解密该对象。下载 SSE-KMS 加密对象时，名为 data-reader 的角色会针对 Amazon S3 在对象上调用 GetObject，并且不会直接调用 Amazon KMS。在收到 GetObject 请求并授权 data-reader 后，Amazon S3 会向 Amazon KMS 发出 FAS 请求以解密 Amazon S3 对象。当 KMS 收到 FAS 请求时，它会检查角色的权限，并且只有在 data-reader 对 KMS 密钥具有正确权限的情况下才会授权解密请求。对 Amazon S3 和 Amazon KMS 的请求均使用角色的权限进行授权，并且只有当 data-reader 同时拥有对 Amazon S3 对象和 Amazon KMS 密钥的权限时，请求才会成功。



> **Note**
>
> 收到 FAS 请求的服务可以发出其他 FAS 请求。在这种情况下，请求的主体必须拥有对 FAS 调用的所有服务的权限。

FAS 请求和 IAM policy 条件

当发出 FAS 请求时，aws:CalledVia、aws:CalledViaFirst 和 aws:CalledViaLast 条件键将填充启动 FAS 调用的服务的服务主体。每当发出 FAS 请求时，aws:ViaAWSService 条件键值都设置为 `true`。在下图中，直接向 CloudFormation 发出的请求未设置任何 `aws:CalledVia` 或 `aws:ViaAWSService` 条件键。当 CloudFormation 和 DynamoDB 代表角色发出下游 FAS 请求时，会填充这些条件键的值。



要在会被具有条件键测试源 IP 地址或源 VPC 的拒绝策略声明拒绝的情况下允许发出 FAS 请求，您必须使用条件键在拒绝策略中为 FAS 请求提供例外。使用 `aws:ViaAWSService` 条件键可以对所有 FAS 请求执行此操作。要仅允许特定 Amazon 服务发出 FAS 请求，请使用 `aws:CalledVia`。

> **Important**
>
> 在通过 VPC 端点发出初始请求后发出 FAS 请求时，来自初始请求的 aws:SourceVpce、aws:SourceVpc 和 aws:VpcSourceIp 的条件键值不用于 FAS 请求中。使用 `aws:VpcSourceIp` 或 `aws:SourceVpce` 编写策略以有条件地授予访问权限时，还必须使用 `aws:ViaAWSService` 或 `aws:CalledVia` 允许 FAS 请求。如果在

公共 Amazon 服务端点收到初始请求后发出 FAS 请求，则后续的 FAS 请求将使用相同的 `aws:SourceIP` 条件键值发出。

示例：允许 Amazon S3 从 VPC 或通过 FAS 进行访问

在以下 IAM policy 示例中，只有当 Amazon S3 GetObject 和 Athena 请求来自附加到 *example_vpc* 的 VPC 端点，或者是 Athena 发出的 FAS 请求时，这些请求才可以被允许。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "OnlyAllowMyIPs",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*",
        "athena:StartQueryExecution",
        "athena:GetQueryResults",
        "athena:GetWorkGroup",
        "athena:StopQueryExecution",
        "athena:GetQueryExecution"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "aws:SourceVPC": [
          "vpc-111bbb22"
          ]
        }
      }
    },
    {
      "Sid": "OnlyAllowFAS",
      "Effect": "Allow",
      "Action": [
        "s3:GetObject*"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "ForAnyValue:StringEquals": {
          "aws:CalledVia": "athena.amazonaws.com"
        }
      }
    }
  ]
}
```

有关使用条件键允许 FAS 访问的其他示例，请参阅数据边界示例策略存储库。

IAM 基于身份的策略示例

策略是 Amazon 中的对象；在与身份或资源相关联时，策略定义它们的权限。在某个 IAM 主体（用户或角色）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略作为附加到 IAM 身份（用户、用户组或角色）的 JSON 文档存储在 Amazon 中。基于身份的策略包括 Amazon 托管策略、客户托管策略和内联策略。要了解如何使用这些示例 JSON 策略文档创建 IAM policy，请参阅。the section called “使用 JSON 编辑器创建策略”

默认情况下，所有请求都会被拒绝，因此必须提供相关权限供身份访问服务、操作和资源。如果还需要允许访问以在 IAM 控制台中完成指定的操作，则需要提供额外的权限。

下面的策略库可以帮助您为自己的 IAM 身份定义权限。在找到所需的策略后，请选择查看该策略以查看该策略的 JSON 版本。您可以将该 JSON 策略文档用作自己策略的模板。

> **Note**
>
> 如果您愿意提交策略供本参考指南采用，请使用该页面底部的 Feedback 按钮。

示例策略：Amazon

- 允许在特定日期范围内进行访问。（查看该策略。）
- 允许启用和禁用 Amazon 区域。（查看该策略。）
- 允许使用 MFA 完成身份验证的用户在安全凭证页面上管理自己的凭证。（查看该策略。）
- 允许在特定日期范围内使用 MFA 进行特定访问。（查看该策略。）
- 允许用户在安全凭证页面上管理自己的凭证。（查看该策略。）
- 允许用户在安全凭证页面上管理自己的 MFA 设备。（查看该策略。）

- 允许用户在安全凭证页面上管理自己的密码。（查看该策略。）
- 允许用户在安全凭证页面上管理自己的密码、访问密钥和 SSH 公有密钥。（查看该策略。）
- 根据请求的区域拒绝对 Amazon 的访问。（查看该策略。）
- 基于源 IP 拒绝对 Amazon 的访问。（查看该策略。）

示例策略：Amazon Data Pipeline

- 拒绝访问不是用户创建的管道（查看该策略。）

示例策略：Amazon DynamoDB

- 允许访问特定的 Amazon DynamoDB 表（查看该策略。）
- 允许访问特定的 Amazon DynamoDB 属性（查看该策略。）
- 允许基于 Amazon Cognito ID 对 Amazon DynamoDB 进行项目级访问（查看该策略。）

示例策略：Amazon EC2

- 允许根据标签为 Amazon EC2 实例附加或分离 Amazon EBS 卷（查看该策略。）
- 允许以编程方式和使用控制台在特定子网中启动 Amazon EC2 实例（查看该策略。）
- 允许以编程方式和在控制台中管理与特定 VPC 关联的 Amazon EC2 安全组（查看该策略。）
- 允许以编程方式和在控制台中启动或停止用户标记的 Amazon EC2 实例（查看该策略。）
- 允许以编程方式和在控制台中根据资源和主体标签启动或停止 Amazon EC2 实例（查看该策略。）
- 在资源和主体标签匹配时，允许启动或停止 Amazon EC2 实例 实例（查看该策略。）
- 允许以编程方式和使用控制台在特定区域进行不受限制的 Amazon EC2 访问（查看该策略。）
- 允许以编程方式和在控制台中启动或停止特定的 Amazon EC2 实例和修改特定的安全组（查看该策略）
- 拒绝在没有 MFA 的情况下访问特定的 Amazon EC2 操作（查看该策略。）
- 将要终止的 Amazon EC2 实例限定为特定的 IP 地址范围（查看该策略）

示例策略：Amazon Identity and Access Management (IAM)

- 允许访问策略模拟器 API（查看该策略。）

- 允许访问策略模拟器控制台（查看该策略。）
- 允许以编程方式和在控制台中担任具有特定标签的任何角色（查看该策略。）
- 允许和拒绝以编程方式和在控制台中访问多个服务（查看该策略。）
- 允许以编程方式和在控制台中为具有不同的特定标签的 IAM 用户添加特定的标签（查看该策略。）
- 允许以编程方式和在控制台中为任何 IAM 用户或角色添加特定的标签（查看该策略。）
- 允许创建仅具有特定标签的新用户（查看该策略。）
- 允许生成和检索 IAM 凭证报告（查看该策略。）
- 允许以编程方式和在控制台中管理组的成员资格（查看该策略。）
- 允许管理特定的标签（查看该策略。）
- 允许将 IAM 角色传递到特定的服务（查看该策略。）
- 允许对 IAM 控制台进行只读访问而不生成报告（查看该策略。）
- 允许对 IAM 控制台进行只读访问（查看该策略。）
- 允许特定用户以编程方式和在控制台中管理组（查看该策略。）
- 允许以编程方式和在控制台中设置账户密码要求（查看该策略。）
- 允许具有特定路径的用户使用策略模拟器 API（查看该策略。）
- 允许具有特定路径的用户使用策略模拟器控制台（查看该策略。）
- 允许 IAM 用户自行管理 MFA 设备（查看该策略。）
- 允许 IAM 用户通过编程方式和控制台设置自己的凭证。（查看该策略。）
- 允许在 IAM 控制台中查看 Amazon Organizations 策略的上次访问的服务信息。（查看该策略。）
- 对可应用于 IAM 用户、组或角色的托管策略加以限制（查看该策略。）

示例策略：Amazon Lambda

- 允许 Amazon Lambda 函数访问 Amazon DynamoDB 表（查看该策略。）

示例策略：Amazon RDS

- Amazon RDS 允许在特定区域中进行完全 RDS 数据库访问。（查看该策略。）
- 允许以编程方式和在控制台中还原 Amazon RDS 数据库（查看该策略。）
- 允许标签所有者不受限制地访问其标记的 Amazon RDS 资源（查看该策略。）

示例策略：Amazon S3

- 允许 Amazon Cognito 用户访问自己的 Amazon S3 存储桶中的对象（查看该策略。）
- 允许具有临时凭证的用户以编程方式和在控制台中访问自己位于 Amazon S3 中的主目录（查看该策略。）
- 允许完全 S3 访问，但如果管理员在过去的 30 分钟内未使用 MFA 登录，则显式拒绝访问 Production 存储桶（查看该策略。）
- 允许 IAM 用户以编程方式和在控制台中访问自己位于 Amazon S3 中的主目录（查看该策略。）
- 允许用户管理单个 Amazon S3 存储桶并拒绝所有其他 Amazon 操作和资源（查看该策略。）
- 允许对特定的 Amazon S3 存储桶进行 `Read` 和 `Write` 访问（查看该策略）
- 允许以编程方式和在控制台中对特定的 Amazon S3 存储桶进行 `Read` 和 `Write` 访问（查看该策略。）

Amazon：允许基于日期和时间进行访问

此示例说明如何创建允许访问基于日期和时间的操作的基于身份的策略。此策略限制访问 2020 年 4 月 1 日和 2020 年 6 月 30 日（含这两个日期）之间发生的操作。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

要了解在 IAM policy 的 `Condition` 块中使用多个条件的信息，请参阅一个条件中包含多个值。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "service-prefix:action-name",
            "Resource": "*",
            "Condition": {
                "DateGreaterThan": {"aws:CurrentTime": "2020-04-01T00:00:00Z"},
                "DateLessThan": {"aws:CurrentTime": "2020-06-30T23:59:59Z"}
            }
        }
    ]
```

```
}
```

> **Note**
>
> 您不能对策略变量使用 Date 条件运算符。要了解更多信息，请参阅 条件元素。

Amazon：允许启用和禁用 Amazon 区域

此示例现在可以创建基于身份的策略，以允许管理员启用和禁用亚太地区（香港）区域（ap-east-1）。此策略定义了程序访问和控制台访问的权限。此设置显示在 Amazon Web Services 管理控制台 中的 Account settings（账户设置）页面。该页面包含只应由账户管理员查看和管理的敏感账户级信息。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

> **Important**
>
> 您无法启用或禁用默认启用的区域。您只能包含默认禁用 的区域。有关更多信息，请参阅《Amazon Web Services 一般参考》中的管理 Amazon 区域。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "EnableDisableHongKong",
            "Effect": "Allow",
            "Action": [
                "account:EnableRegion",
                "account:DisableRegion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {"account:TargetRegion": "ap-east-1"}
            }
        },
        {
```

```
            "Sid": "ViewConsole",
            "Effect": "Allow",
            "Action": [
                "account:ListRegions"
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。

此示例演示了如何创建基于身份的策略，以允许通过多重身份验证（MFA）完成身份验证的 IAM 用户在安全凭证页面上管理自己的凭证。此 Amazon Web Services 管理控制台 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、MFA 设备、X.509 证书、SSH 密钥和 Git 凭证。此示例策略包括查看和编辑页面上所有信息所需的权限。它还要求用户在执行 Amazon 中的任何其他操作之前使用 MFA 进行设置和身份验证。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的凭证。

要了解用户如何访问安全凭证页面，请参阅 IAM 用户如何更改自己的密码（控制台）。

> **Note**
>
> - 此示例策略不允许用户在首次登录 Amazon Web Services 管理控制台 的同时重置密码。我们建议您在新用户登录之前不要向他们授予权限。有关更多信息，请参阅 如何安全地创建 IAM 用户？。这还可以防止密码过期的用户在登录期间重置其密码。您可以通过向语句 `DenyAllExceptListedIfNoMFA` 中添加 `iam:ChangePassword` 和 `iam:GetAccountPasswordPolicy` 来允许此操作。但是，我们不建议您这样做，因为允许用户在未进行 MFA 验证的情况下更改密码可能存在安全风险。
> - 如果您打算将此策略用于编程访问，则必须调用 `GetSessionToken` 以使用 MFA 进行身份验证。有关更多信息，请参阅 使用 MFA 保护 API 访问。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 `"Resource" : "*"`。此语句包括允许用户查看特定信息的以下操作：
  - `GetAccountPasswordPolicy` – 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
  - `ListVirtualMFADevices` - 查看有关为用户启用的虚拟 MFA 设备的详细信息。
- `AllowManageOwnPasswords` 语句还允许用户更改他们自己的密码。此语句还包括 `GetUser` 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- `AllowManageOwnAccessKeys` 语句允许用户创建、更新和删除他们自己的访问密钥。用户还可以检索有关上次使用指定访问密钥的时间信息。
- `AllowManageOwnSigningCertificates` 语句允许用户上传、更新和删除他们自己的签名证书。
- `AllowManageOwnSSHPublicKeys` 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。
- `AllowManageOwnGitCredentials` 语句允许用户创建、更新和删除他们自己的 CodeCommit 的 Git 凭证。
- `AllowManageOwnVirtualMFADevice` 语句允许用户创建他们自己的虚拟 MFA 设备。此语句中的资源 ARN 允许用户创建任何名称的 MFA 设备，但策略中的其他语句仅允许用户将设备连接到当前登录的用户。
- `AllowManageOwnUserMFA` 语句允许用户查看或管理他们自己用户的虚拟、U2F 或硬件 MFA 设备。此语句中的资源 ARN 仅允许访问用户自己的 IAM 用户。用户无法查看或管理其他用户的 MFA 设备。
- `DenyAllExceptListedIfNoMFA` 语句拒绝访问所有 Amazon 服务中的每个操作（除了一些列出的操作），但前提是用户未使用 MFA 登录。该语句使用 `"Deny"` 和 `"NotAction"` 的组合来显式拒绝对未列出的每个操作的访问。此语句不会拒绝或允许列出的项目。但是，策略中的其他语句允许这些操作。有关此语句的逻辑的更多信息，请参阅 NotAction 以及 Deny。如果用户已使用 MFA 登录，则 `Condition` 测试将失败，并且此语句不会拒绝任何操作。在这种情况下，用户的其他策略或语句确定用户的权限。

  此语句确保当用户未使用 MFA 登录时，他们只能执行列出的操作。此外，只有在另一个语句或策略允许访问这些操作时，它们才能执行列出的操作。这不允许用户在登录时创建密码，因为如果没有 MFA 授权，则不允许 `iam:ChangePassword` 操作。

  `...IfExists` 运算符的 `Bool` 版本可确保：如果 `aws:MultiFactorAuthPresent` 键缺失，条件将返回 true。这意味着使用长期凭证（例如访问密钥）访问 API 的用户被拒绝访问非 IAM API 操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 AllowViewAccountInfo 和 DenyAllExceptListedIfNoMFA 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 iam:GetLoginProfile 和 iam:UpdateLoginProfile 操作添加到 AllowManageOwnPasswords 语句。要允许用户在不使用 MFA 登录的情况下从自己的用户页面更改密码，请将 iam:UpdateLoginProfile 操作添加到 DenyAllExceptListedIfNoMFA 语句。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
```

```
        {
            "Sid": "AllowManageOwnSigningCertificates",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSigningCertificate",
                "iam:ListSigningCertificates",
                "iam:UpdateSigningCertificate",
                "iam:UploadSigningCertificate"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnGitCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:ResetServiceSpecificCredential",
                "iam:UpdateServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
```

```
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

Amazon：允许在特定日期内使用 MFA 进行特定访问

此示例说明了如何创建一个基于身份的策略，该策略将使用多个条件，借助 AND 逻辑对这些条件进行评估。它允许对名为 SERVICE-NAME-1 的服务进行完全访问，并且允许对名为 ACTION-NAME-A 的服务中的 ACTION-NAME-B 和 SERVICE-NAME-2 操作进行访问。只有在用户使用多重身份验证 (MFA) 进行身份验证时，才允许执行这些操作。只能对在 2017 年 7 月 1 日至 2017 年 12 月 31 日 (UTC 时间，包含这两个时间) 期间发生的操作进行访问。此策略授予有计划地通过 Amazon API 或

Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

要了解在 IAM policy 的 `Condition` 块中使用多个条件的信息，请参阅一个条件中包含多个值。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "service-prefix-1:*",
            "service-prefix-2:action-name-a",
            "service-prefix-2:action-name-b"
        ],
        "Resource": "*",
        "Condition": {
            "Bool": {"aws:MultiFactorAuthPresent": true},
            "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
            "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
        }
    }
}
```

Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的凭证

此示例演示了如何创建基于身份的策略，以允许 IAM 用户在安全凭证页面上管理自己的所有凭证。此 Amazon Web Services 管理控制台 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、X.509 证书、SSH 密钥和 Git 凭证。此示例策略包括查看和编辑页面上所有信息（用户的 MFA 设备除外）所需的权限。要允许用户使用 MFA 管理他们自己的凭证，请参阅Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。

要了解用户如何访问安全凭证页面，请参阅 IAM 用户如何更改自己的密码（控制台）。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 `"Resource" : "*"`。此语句包括允许用户查看特定信息的以下操作：

  - GetAccountPasswordPolicy – 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
  - GetAccountSummary - 查看账户 ID 和账户的规范用户 ID。
- AllowManageOwnPasswords 语句还允许用户更改他们自己的密码。此语句还包括 GetUser 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- AllowManageOwnAccessKeys 语句允许用户创建、更新和删除他们自己的访问密钥。用户还可以检索有关上次使用指定访问密钥的时间信息。
- AllowManageOwnSigningCertificates 语句允许用户上传、更新和删除他们自己的签名证书。
- AllowManageOwnSSHPublicKeys 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。
- AllowManageOwnGitCredentials 语句允许用户创建、更新和删除他们自己的 CodeCommit 的 Git 凭证。

该策略不允许用户查看或管理他们自己的 MFA 设备。他们也不能在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 iam:ListUsers 操作添加到 AllowViewAccountInfo 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 iam:CreateLoginProfile、iam:DeleteLoginProfile、iam:GetLoginProfile 和 iam:UpdateLoginProfile 操作添加到 AllowManageOwnPasswords 语句。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
```

```
            "iam:ChangePassword",
            "iam:GetUser"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnAccessKeys",
        "Effect": "Allow",
        "Action": [
            "iam:CreateAccessKey",
            "iam:DeleteAccessKey",
            "iam:ListAccessKeys",
            "iam:UpdateAccessKey",
            "iam:GetAccessKeyLastUsed"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnSigningCertificates",
        "Effect": "Allow",
        "Action": [
            "iam:DeleteSigningCertificate",
            "iam:ListSigningCertificates",
            "iam:UpdateSigningCertificate",
            "iam:UploadSigningCertificate"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnSSHPublicKeys",
        "Effect": "Allow",
        "Action": [
            "iam:DeleteSSHPublicKey",
            "iam:GetSSHPublicKey",
            "iam:ListSSHPublicKeys",
            "iam:UpdateSSHPublicKey",
            "iam:UploadSSHPublicKey"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}"
    },
    {
        "Sid": "AllowManageOwnGitCredentials",
        "Effect": "Allow",
        "Action": [
```

```
                "iam:CreateServiceSpecificCredential",
                "iam:DeleteServiceSpecificCredential",
                "iam:ListServiceSpecificCredentials",
                "iam:ResetServiceSpecificCredential",
                "iam:UpdateServiceSpecificCredential"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的 MFA 设备。

此示例演示了如何创建基于身份的策略，以允许通过多重身份验证（MFA）完成身份验证的 IAM 用户在安全凭证页面上管理自己的 MFA 设备。此 Amazon Web Services 管理控制台 页面显示账户和用户信息，但是用户只能查看和编辑他们自己的 MFA 设备。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。

> **Note**
>
> 如果具有此策略的 IAM 用户未通过 MFA 进行身份验证，则此策略会拒绝对所有 Amazon 操作（使用 MFA 进行身份验证所需的操作除外）的访问。要使用 Amazon CLI 和 Amazon API，IAM 用户必须先使用 Amazon STS GetSessionToken 操作检索其 MFA 令牌，然后使用该令牌验证所需的操作。其他策略（如基于资源的策略或其他基于身份的策略）可以允许其他服务中的操作。如果 IAM 用户没有经过 MFA 身份验证，则此策略将拒绝该访问。

要了解用户如何访问安全凭证页面，请参阅 IAM 用户如何更改自己的密码（控制台）。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看有关为用户启用的虚拟 MFA 设备的详细信息。此权限必须在它自己的语句中，因为它不支持指定资源 ARN。相反，您必须指定 `"Resource" : "*"`。
- `AllowManageOwnVirtualMFADevice` 语句允许用户创建他们自己的虚拟 MFA 设备。此语句中的资源 ARN 允许用户创建任何名称的 MFA 设备，但策略中的其他语句仅允许用户将设备连接到当前登录的用户。

- `AllowManageOwnUserMFA` 语句允许用户查看或管理他们自己的虚拟、U2F 或硬件 MFA 设备。此语句中的资源 ARN 仅允许访问用户自己的 IAM 用户。用户无法查看或管理其他用户的 MFA 设备。
- `DenyAllExceptListedIfNoMFA` 语句拒绝访问所有 Amazon 服务中的每个操作（除了一些列出的操作），但前提是用户未使用 MFA 登录。该语句使用 "Deny" 和 "NotAction" 的组合来显式拒绝对未列出的每个操作的访问。此语句不会拒绝或允许列出的项目。但是，策略中的其他语句允许这些操作。有关此语句的逻辑的更多信息，请参阅 NotAction 以及 Deny。如果用户已使用 MFA 登录，则 `Condition` 测试将失败，并且此语句不会拒绝任何操作。在这种情况下，用户的其他策略或语句确定用户的权限。

  此语句确保当用户未使用 MFA 登录时，他们只能执行列出的操作。此外，只有在另一个语句或策略允许访问这些操作时，它们才能执行列出的操作。

  `...IfExists` 运算符的 `Bool` 版本可确保：如果 `aws:MultiFactorAuthPresent` 键缺失，条件将返回 true。这意味着使用长期凭证（例如访问密钥）访问 API 操作的用户被拒绝访问非 IAM API 操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 `iam:ListUsers` 操作添加到 `AllowViewAccountInfo` 和 `DenyAllExceptListedIfNoMFA` 语句。

> **Warning**
>
> 在未通过 MFA 进行身份验证的情况下，请不要添加删除 MFA 设备的权限。具有该策略的用户可能会尝试为自己分配一个虚拟 MFA 设备，并出现错误以指示未授权其执行 `iam:DeleteVirtualMFADevice`。如果发生这种情况，请不要将该权限添加到 `DenyAllExceptListedIfNoMFA` 语句中。切勿允许未使用 MFA 进行身份验证的用户删除 MFA 设备。如果用户以前开始将虚拟 MFA 设备分配给其用户并取消了该过程，则可能会看到该错误。要解决该问题，您或其他管理员必须使用 Amazon CLI 或 Amazon API 删除用户的现有虚拟 MFA 设备。有关更多信息，请参阅 我没有权限执行：iam:DeleteVirtualMFADevice。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": "iam:ListVirtualMFADevices",
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowManageOwnUserMFA",
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "DenyAllExceptListedIfNoMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:GetUser",
                "iam:ListMFADevices",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice",
                "sts:GetSessionToken"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}
            }
        }
    ]
```

```
}
```

Amazon：允许 IAM 用户在的“安全凭证”页面上更改自己的控制台密码

此示例演示了如何创建基于身份的策略，以允许 IAM 用户在安全凭证页面上更改其 Amazon Web Services 管理控制台 密码。此 Amazon Web Services 管理控制台 页面显示账户和用户信息，但是用户只能访问他们自己的密码。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的凭证。

要了解用户如何访问安全凭证页面，请参阅 IAM 用户如何更改自己的密码（控制台）。

此策略有何作用？

- `ViewAccountPasswordRequirements` 语句允许用户查看账户密码要求，同时更改他们自己的 IAM 用户密码。
- `ChangeOwnPassword` 语句还允许用户更改他们自己的密码。此语句还包括 `GetUser` 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。

此策略不允许用户在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 `iam:ListUsers` 操作添加到 `ViewAccountPasswordRequirements` 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 `iam:GetLoginProfile` 和 `iam:UpdateLoginProfile` 操作添加到 `ChangeOwnPassword` 语句。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewAccountPasswordRequirements",
            "Effect": "Allow",
            "Action": "iam:GetAccountPasswordPolicy",
            "Resource": "*"
        },
        {
            "Sid": "ChangeOwnPassword",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:GetUser",
                "iam:ChangePassword"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的密码、访问密钥和 SSH 公有密钥

此示例演示了如何创建基于身份的策略，以允许 IAM 用户在安全凭证页面上管理其密码、访问密钥和 X.509 证书。此 Amazon Web Services 管理控制台 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、MFA 设备、X.509 证书、SSH 密钥和 Git 凭证。此示例策略包括仅查看和编辑其密码、访问密钥和 X.509 证书所需的权限。要允许用户使用 MFA 管理他们自己的所有凭证，请参阅Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。要允许用户在不使用 MFA 的情况下管理自己的凭证，请参阅Amazon：允许 IAM 用户在“安全凭证”页面上管理自己的凭证。

要了解用户如何访问安全凭证页面，请参阅 IAM 用户如何更改自己的密码（控制台）。

此策略有何作用？

- `AllowViewAccountInfo` 语句允许用户查看账户级信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 `"Resource" : "*"`。此语句包括允许用户查看特定信息的以下操作：
  - `GetAccountPasswordPolicy` – 查看账户密码要求，同时更改他们自己的 IAM 用户密码。
  - `GetAccountSummary` - 查看账户 ID 和账户的规范用户 ID。
- `AllowManageOwnPasswords` 语句还允许用户更改他们自己的密码。此语句还包括 `GetUser` 操作，查看 My Security Credentials（我的安全凭证）页面上的大多数信息都需要此操作。
- `AllowManageOwnAccessKeys` 语句允许用户创建、更新和删除他们自己的访问密钥。用户还可以检索有关上次使用指定访问密钥的时间信息。
- `AllowManageOwnSSHPublicKeys` 语句允许用户上传、更新和删除他们自己的 CodeCommit 的 SSH 公有密钥。

该策略不允许用户查看或管理他们自己的 MFA 设备。他们也不能在 IAM 控制台中查看 Users（用户）页面，或使用该页面访问自己的用户信息。要允许此操作，请将 `iam:ListUsers` 操作添加到 `AllowViewAccountInfo` 语句。它也不允许用户在自己的用户页面上更改密码。要允许此操作，请将 `iam:GetLoginProfile` 和 `iam:UpdateLoginProfile` 操作添加到 `AllowManageOwnPasswords` 语句。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewAccountInfo",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccountPasswordPolicy",
                "iam:GetAccountSummary"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowManageOwnPasswords",
            "Effect": "Allow",
            "Action": [
                "iam:ChangePassword",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowManageOwnAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:CreateAccessKey",
                "iam:DeleteAccessKey",
                "iam:ListAccessKeys",
                "iam:UpdateAccessKey",
                "iam:GetAccessKeyLastUsed"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
```

```
            "Sid": "AllowManageOwnSSHPublicKeys",
            "Effect": "Allow",
            "Action": [
                "iam:DeleteSSHPublicKey",
                "iam:GetSSHPublicKey",
                "iam:ListSSHPublicKeys",
                "iam:UpdateSSHPublicKey",
                "iam:UploadSSHPublicKey"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

Amazon：根据请求的区域拒绝访问 Amazon

此示例说明了如何创建一个基于身份的策略，以拒绝对使用 aws:RequestedRegion 条件键指定的区域以外的任何操作的访问，但服务中使用 NotAction 指定的操作除外。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的#######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

该策略使用具有 Deny 效果的 NotAction 元素，这会显式拒绝访问语句中未 列出的所有操作。不应拒绝 CloudFront、IAM、Route 53 和 Amazon Web Services 支持 服务中的操作，因为这些服务是常用的 Amazon 全球服务，并具有一个实际位于 us-east-1 区域的终端节点。由于针对这些服务的所有请求都是向 us-east-1 区域发出的，因此，将在没有 NotAction 元素的情况下拒绝这些请求。请编辑该元素以包含您使用的其他 Amazon 全球服务（如 budgets、globalaccelerator、importexport、organizations 或 waf）的操作。一些其他全球服务（例如，聊天应用程序中的 Amazon Q 开发者版和 Amazon Device Farm）是具有物理位置位于 us-west-2 区域的终端节点的全球服务。要了解具有单个全球端点的所有服务，请参阅《Amazon Web Services 一般参考》中的 Amazon 区域和端点。有关使用具有 Deny 效果的 NotAction 元素的更多信息，请参阅IAM JSON 策略元素：NotAction。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "DenyAllOutsideRequestedRegions",
            "Effect": "Deny",
            "NotAction": [
                "cloudfront:*",
                "iam:*",
                "organizations:*",
                "route53:*",
                "support:*"
            ],
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:RequestedRegion": [
                        "eu-central-1",
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

Amazon：基于源 IP 拒绝对 Amazon 的访问

此示例说明了如何创建基于身份的策略，以当请求来自指定 IP 范围以外的主体时拒绝对账户中所有 Amazon 操作的访问。当您公司的 IP 地址位于指定范围内时，该策略很有用。在此示例中，除非源自 CIDR 范围 192.0.2.0/24 或 203.0.113.0/24, 否则请求将被拒绝。该策略不拒绝使用 转发访问会话 的 Amazon 服务发出的请求，因为原始请求者的 IP 地址已保留。

在与 `"Effect": "Deny"` 相同的策略语句中谨慎使用否定条件。使用否定条件时，在所有条件下（指定的条件除外），在策略语句中指定的操作将被明确拒绝。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

当其他策略允许操作时，主体可以从 IP 地址范围内发出请求。Amazon 服务还可以使用主体的凭证发出请求。当主体从 IP 范围之外发出请求时，请求将被拒绝。

有关使用 `aws:SourceIp` 条件键的更多信息，包括有关 `aws:SourceIp` 可能无法在您的策略中起作用的信息，请参阅 Amazon 全局条件上下文密钥。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Deny",
        "Action": "*",
        "Resource": "*",
        "Condition": {
            "NotIpAddress": {
                "aws:SourceIp": [
                    "192.0.2.0/24",
                    "203.0.113.0/24"
                ]
            }
        }
    }
}
```

Amazon：拒绝访问您账户之外的 Amazon S3 资源，Amazon Web Services Data Exchange 除外。

以下示例说明如何创建基于身份的策略以拒绝访问 Amazon 中不属于您的账户的所有资源，Amazon Web Services Data Exchange 正常操作所需的资源除外。要使用此策略，请将示例策略中的###### #替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

您可以使用条件键 Amazon Web Services Data Exchange 和 `aws:ResourceOrgPaths` 来创建类似的策略，限制对企业或企业部门内部资源的访问，同时包括 `aws:ResourceOrgID` 所拥有的资源。

如果您在您的环境中使用 Amazon Web Services Data Exchange，该服务会创建资源并与之交互，例如服务账户拥有的 Amazon S3 存储桶。例如，Amazon Web Services Data Exchange 代表 IAM 主体（用户或角色）向 Amazon Web Services Data Exchange 服务拥有的 Amazon S3 存储桶发送调用 Amazon Web Services Data Exchange API 的请求。在这种情况下，在策略中使用 `aws:ResourceAccount`、`aws:ResourceOrgPaths` 或 `aws:ResourceOrgID`（不包括 Amazon Web Services Data Exchange 所拥有的资源）会拒绝对服务账户拥有的存储桶的访问。

- 语句 `DenyAllAwsResourcesOutsideAccountExceptS3` 使用带有 Deny 效果的 `NotAction` 元素，它显式拒绝访问未在声明中列出和不属于已列出账户的所有操作。`NotAction` 元素表示此语句的例外情况。这些操作是此语句的例外情况，因为如果这些操作是在 Amazon Web Services Data Exchange 创建的资源上执行的，则策略会拒绝这些操作。
- 语句 `DenyAllS3ResourcesOutsideAccountExceptDataExchange` 使用 `ResourceAccount` 和 `CalledVia` 条件的组合，拒绝访问前一语句中排除的三个 Amazon SNS 操作。如果资源不属于已列出的账户，并且，如果调用服务的不是 Amazon Web Services Data Exchange，则该语句会拒绝这些操作。如果资源属于列出的账户，或由列出的服务主体 `dataexchange.amazonaws.com` 执行操作，则该语句不会拒绝这些操作。

> **Important**
>
> 该策略不允许进行任何操作。该策略使用 Deny 效果，这会显式拒绝访问语句中列出而不属于已列出账户的所有资源。将此策略与允许访问特定资源的其他策略结合使用。

以下示例说明如何配置策略以允许访问所需的 Amazon S3 存储桶。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAllAwsReourcesOutsideAccountExceptAmazonS3",
      "Effect": "Deny",
      "NotAction": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
```

```
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    },
    {
      "Sid": "DenyAllS3ResourcesOutsideAccountExceptDataExchange",
      "Effect": "Deny",
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:PutObjectAcl"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        },
        "ForAllValues:StringNotEquals": {
          "aws:CalledVia": [
            "dataexchange.amazonaws.com"
          ]
        }
      }
    }
  ]
}
```

Amazon Data Pipeline：拒绝用户访问他人创建的 DataPipeline 管道

此示例说明如何创建基于身份的策略以拒绝对用户未创建的管道的访问。如果 `PipelineCreator` 字段的值与 IAM 用户名匹配，则指定的操作不会被拒绝。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExplicitDenyIfNotTheOwner",
            "Effect": "Deny",
            "Action": [
                "datapipeline:ActivatePipeline",
                "datapipeline:AddTags",
                "datapipeline:DeactivatePipeline",
                "datapipeline:DeletePipeline",
                "datapipeline:DescribeObjects",
                "datapipeline:EvaluateExpression",
                "datapipeline:GetPipelineDefinition",
                "datapipeline:PollForTask",
                "datapipeline:PutPipelineDefinition",
                "datapipeline:QueryObjects",
                "datapipeline:RemoveTags",
                "datapipeline:ReportTaskProgress",
                "datapipeline:ReportTaskRunnerHeartbeat",
                "datapipeline:SetStatus",
                "datapipeline:SetTaskStatus",
                "datapipeline:ValidatePipelineDefinition"
            ],
            "Resource": ["*"],
            "Condition": {
                "StringNotEquals": {"datapipeline:PipelineCreator":
 "${aws:userid}"}
            }
        }
    ]
}
```

Amazon DynamoDB：允许访问特定的表

此示例说明了如何创建基于身份的策略以允许对 MyTable DynamoDB 表的完全访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

> **Important**
>
> 该策略允许可在 DynamoDB 表上执行的所有操作。要了解这些操作的信息，请参阅 Amazon DynamoDB 开发人员指南中的 DynamoDB API 权限：操作、资源和条件参考。您可以通过列出每个单独的操作来提供同样的权限。不过，如果您在 Action 元素中使用通配符 (*)（例如，"dynamodb:List*"），则在 DynamoDB 添加新的列表操作时无需更新策略。

该策略只允许在具有指定名称的 DynamoDB 表上执行操作。要允许您的用户对 DynamoDB 中的所有内容进行 Read 访问，也可以附加 AmazonDynamoDBReadOnlyAccess Amazon 托管策略。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListAndDescribe",
            "Effect": "Allow",
            "Action": [
                "dynamodb:List*",
                "dynamodb:DescribeReservedCapacity*",
                "dynamodb:DescribeLimits",
                "dynamodb:DescribeTimeToLive"
            ],
            "Resource": "*"
        },
        {
            "Sid": "SpecificTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGet*",
                "dynamodb:DescribeStream",
                "dynamodb:DescribeTable",
                "dynamodb:Get*",
```

```
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWrite*",
                "dynamodb:CreateTable",
                "dynamodb:Delete*",
                "dynamodb:Update*",
                "dynamodb:PutItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/MyTable"
        }
    ]
}
```

Amazon DynamoDB：允许访问特定的属性

此示例说明如何创建基于身份的策略以允许访问特定 DynamoDB 属性。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

`dynamodb:Select` 要求会防止 API 操作返回未授权的属性，例如来自索引投影的属性。要了解有关 DynamoDB 条件键的更多信息，请参阅 Amazon DynamoDB 开发人员指南中的指定条件：使用条件键。要了解在 IAM policy 的 `Condition` 块中使用多个条件或多个条件键的信息，请参阅一个条件中包含多个值。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:GetItem",
                "dynamodb:BatchGetItem",
                "dynamodb:Query",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem",
                "dynamodb:DeleteItem",
                "dynamodb:BatchWriteItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/table-name"],
```

```
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:Attributes": [
                        "column-name-1",
                        "column-name-2",
                        "column-name-3"
                    ]
                },
                "StringEqualsIfExists": {"dynamodb:Select":
 "SPECIFIC_ATTRIBUTES"}
            }
        }
    ]
}
```

Amazon DynamoDB：允许基于 Amazon Cognito ID 对 DynamoDB 进行项目级别的访问

此示例演示了如何创建基于身份的策略，以根据 Amazon Cognito 身份池用户 ID 授予对 `MyTable` DynamoDB 表的项目级访问权限。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

要使用此策略，DynamoDB 表的结构必须将 Amazon Cognito 身份池用户 ID 作为分区键。有关更多信息，请参阅 Amazon DynamoDB 开发人员指南中的创建表。

要了解有关 DynamoDB 条件键的更多信息，请参阅 Amazon DynamoDB 开发人员指南中的指定条件：使用条件键。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
```

```
                "dynamodb:Query",
                "dynamodb:UpdateItem"
            ],
            "Resource": ["arn:aws:dynamodb:*:*:table/MyTable"],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": ["${cognito-
identity.amazonaws.com:sub}"]
                }
            }
        }
    ]
}
```

Amazon EC2：基于标签为 EC2 实例附加或分离 Amazon EBS 卷

此示例显示您可以创建基于身份的策略，允许 EBS 卷所有者为标记为开发实例（`VolumeUser`）的 EC2 实例附加或分离使用标签 `Department=Development` 定义的 EBS 卷。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

有关创建 IAM 策略以控制对 Amazon EC2 资源访问的更多信息，请参阅《Amazon EC2 用户指南》中的控制对 Amazon EC2 资源的访问。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/Department": "Development"}
            }
        },
        {
```

```
            "Effect": "Allow",
            "Action": [
                "ec2:AttachVolume",
                "ec2:DetachVolume"
            ],
            "Resource": "arn:aws:ec2:*:*:volume/*",
            "Condition": {
                "StringEquals": {"aws:ResourceTag/VolumeUser": "${aws:username}"}
            }
        }
    ]
}
```

Amazon EC2：允许以编程方式和使用控制台在特定子网中启动 EC2 实例

此示例说明您如何创建基于身份的策略以允许列出所有 EC2 对象的信息并在特定的子网中启动 EC2 实例。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:GetConsole*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "ec2:RunInstances",
            "Resource": [
                "arn:aws:ec2:*:*:subnet/subnet-subnet-id",
                "arn:aws:ec2:*:*:network-interface/*",
                "arn:aws:ec2:*:*:instance/*",
                "arn:aws:ec2:*:*:volume/*",
                "arn:aws:ec2:*::image/ami-*",
```

```
                "arn:aws:ec2:*:*:key-pair/*",
                "arn:aws:ec2:*:*:security-group/*"
            ]
        }
    ]
}
```

Amazon EC2：允许以编程方式和在控制台中管理关联至特定标签密钥值对的 EC2 安全组

此示例说明如何创建基于身份的策略，该策略授予用户对具有相同标签的安全组采取某些操作的权限。该策略授予相应的权限，以便在 Amazon EC2 控制台中查看安全组、添加和删除入站和出站规则，以及为具有标签 `Department=Test` 的现有安全组列出并修改规则描述。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
      "Effect": "Allow",
      "Action": [
         "ec2:DescribeSecurityGroups",
         "ec2:DescribeSecurityGroupRules",
         "ec2:DescribeTags"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
         "ec2:AuthorizeSecurityGroupIngress",
         "ec2:RevokeSecurityGroupIngress",
         "ec2:AuthorizeSecurityGroupEgress",
         "ec2:RevokeSecurityGroupEgress",
         "ec2:ModifySecurityGroupRules",
         "ec2:UpdateSecurityGroupRuleDescriptionsIngress",
         "ec2:UpdateSecurityGroupRuleDescriptionsEgress"
      ],
```

```
      "Resource": [
         "arn:aws:ec2:us-east-1:111122223333:security-group/*"
      ],
      "Condition": {
         "StringEquals": {
            "aws:ResourceTag/Department": "Test"
         }
      }
     },
     {
      "Effect": "Allow",
      "Action": [
         "ec2:ModifySecurityGroupRules"
      ],
      "Resource": [
         "arn:aws:ec2:us-east-1:111122223333:security-group-rule/*"
      ]
     }
   ]
}
```

Amazon EC2：允许以编程方式和在控制台中启动或停止用户已标记的 EC2 实例

此示例说明了如何创建基于身份的策略以允许 IAM 用户启动或停止 EC2 实例，但仅限实例标签 `Owner` 具有该用户的用户名值时。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
            "Resource": "arn:aws:ec2:*:*:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Owner": "${aws:username}"
```

```
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "ec2:DescribeInstances",
            "Resource": "*"
        }
    ]
}
```

EC2：基于标签来启动或停止实例

此示例说明了如何创建基于身份的策略以允许使用标签键值对 `Project = DataAnalytics` 启动或停止实例，但仅限具有标签键值对 `Department = Data` 的主体执行。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

如果策略中的条件的两个部分都为 true，则此条件返回 true。实例必须具有 `Project=DataAnalytics` 标签。此外，发起请求的 IAM 主体（用户或角色）必须具有 `Department=Data` 标签。

> **Note**
>
> 作为最佳实践，请将具有 `aws:PrincipalTag` 条件键的策略附加到 IAM 组，以应对一些用户可能具有指定标签而另一些用户可能没有这些标签的情况。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "StartStopIfTags",
            "Effect": "Allow",
            "Action": [
                "ec2:StartInstances",
                "ec2:StopInstances"
            ],
```

```
            "Resource": "arn:aws:ec2:us-east-1:123456789012:instance/*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/Project": "DataAnalytics",
                    "aws:PrincipalTag/Department": "Data"
                }
            }
        }
    ]
}
```

EC2：基于匹配的主体和资源标签来启动或停止实例

此示例说明如何创建基于身份的策略以允许主体在实例的资源标签和主体标签具有相同的标签键值 `CostCenter` 时启动或停止 Amazon EC2 实例。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

> **Note**
>
> 作为最佳实践，请将具有 `aws:PrincipalTag` 条件键的策略附加到 IAM 组，以应对一些用户可能具有指定标签而另一些用户可能没有这些标签的情况。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "ec2:startInstances",
            "ec2:stopInstances"
        ],
        "Resource": "*",
        "Condition": {"StringEquals":
            {"aws:ResourceTag/CostCenter": "${aws:PrincipalTag/CostCenter}"}}
    }
}
```

Amazon EC2：允许以编程方式和使用控制台在特定区域进行不受限制的 EC2 访问

此示例说明了如何创建基于身份的策略以允许在特定区域中进行完全 EC2 访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。有关区域代码的列表，请参阅 Amazon EC2 用户指南中的可用区域。

或者，您也可以使用全局条件键 `aws:RequestedRegion`，所有 Amazon EC2 API 操作均支持该条件键。有关更多信息，请参阅 Amazon EC2 用户指南中的示例：限制对特定区域的访问。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": "ec2:*",
            "Resource": "*",
            "Effect": "Allow",
            "Condition": {
                "StringEquals": {
                    "ec2:Region": "us-east-2"
                }
            }
        }
    ]
}
```

Amazon EC2：允许以编程方式和在控制台中启动或停止 EC2 实例并修改安全组

此示例说明如何创建基于身份的策略以允许启动或停止特定的 EC2 实例并修改特定的安全组。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Action": [
        "ec2:DescribeInstances",
        "ec2:DescribeSecurityGroups",
        "ec2:DescribeSecurityGroupReferences",
        "ec2:DescribeStaleSecurityGroups"
      ],
      "Resource": "*",
      "Effect": "Allow"
    },
    {
      "Action": [
        "ec2:AuthorizeSecurityGroupEgress",
        "ec2:AuthorizeSecurityGroupIngress",
        "ec2:RevokeSecurityGroupEgress",
        "ec2:RevokeSecurityGroupIngress",
        "ec2:StartInstances",
        "ec2:StopInstances"
      ],
      "Resource": [
        "arn:aws:ec2:*:*:instance/i-instance-id",
        "arn:aws:ec2:*:*:security-group/sg-security-group-id"
      ],
      "Effect": "Allow"
    }
  ]
}
```

Amazon EC2：要求进行 MFA (GetSessionToken) 以执行特定的 EC2 操作

此示例说明如何创建基于身份的策略以允许完全访问 Amazon EC2 中的 Amazon API 操作。不过，如果未使用多重身份验证 (MFA) 验证用户身份，则它显式拒绝访问 `StopInstances` 和 `TerminateInstances` API 操作。要以编程方式执行该操作，用户必须在调用 `GetSessionToken` 操作时包含可选的 `SerialNumber` 和 `TokenCode` 值。该操作返回已使用 MFA 验证的临时凭证。要了解 GetSessionToken 的更多信息，请参阅为不受信任的环境中的用户请求凭证。

此策略有何作用？

- `AllowAllActionsForEC2` 语句允许所有 Amazon EC2 操作。

- 在缺少 MFA 上下文时，DenyStopAndTerminateWhenMFAIsNotPresent 语句拒绝 StopInstances 和 TerminateInstances 操作。这意味着，在缺少多重身份验证上下文（表示不使用 MFA）时，将拒绝这些操作。拒绝将覆盖允许。

> **Note**
>
> Deny 语句中的 MultiFactorAuthPresent 的条件检查不应为 {"Bool": {"aws:MultiFactorAuthPresent":false}}，因为在未使用 MFA 时，该键不存在并且无法进行评估。因此，在检查值之前，请使用 BoolIfExists 检查以确定该键是否存在。有关更多信息，请参阅 ...IfExists 条件运算符。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllActionsForEC2",
            "Effect": "Allow",
            "Action": "ec2:*",
            "Resource": "*"
        },
        {
            "Sid": "DenyStopAndTerminateWhenMFAIsNotPresent",
            "Effect": "Deny",
            "Action": [
                "ec2:StopInstances",
                "ec2:TerminateInstances"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {"aws:MultiFactorAuthPresent": false}
            }
        }
    ]
}
```

Amazon EC2：将要终止的 EC2 实例限制为某个 IP 地址范围

此示例显示了如果请求来自指定 IP 范围以外，您可以创建基于身份的策略，通过允许操作但显式拒绝访问来限制 EC2 实例。当您公司的 IP 地址位于指定范围内时，该策略很有用。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

如果将该策略与允许 `ec2:TerminateInstances` 操作的其他策略（例如 AmazonEC2FullAccess Amazon 托管策略）一起使用，则访问会被拒绝。这是因为“显式拒绝”声明优先于“允许”声明。有关更多信息，请参阅 Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。

> **Important**
>
> `aws:SourceIp` 条件键拒绝对 Amazon Web Services 服务（例如 Amazon CloudFormation）的访问（代表您进行的调用）。有关使用 `aws:SourceIp` 条件键的更多信息，请参阅Amazon 全局条件上下文密钥。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["ec2:TerminateInstances"],
            "Resource": ["*"]
        },
        {
            "Effect": "Deny",
            "Action": ["ec2:TerminateInstances"],
            "Condition": {
                "NotIpAddress": {
                    "aws:SourceIp": [
                        "192.0.2.0/24",
                        "203.0.113.0/24"
                    ]
                }
            },
            "Resource": ["*"]
        }
```

```
    ]
}
```

IAM：访问策略模拟器 API

此示例说明您如何创建基于身份的策略，允许对附加到当前 Amazon Web Services 账户 中的用户、组或角色的策略使用策略模拟器 API。该策略还允许模拟以字符串形式传递给该 API 的不太敏感的策略。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForCustomPolicy",
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulateCustomPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 要允许用户访问策略模拟器控制台，以模拟附加到当前 Amazon Web Services 账户 中用户、组或角色的策略，请参阅IAM：访问策略模拟器控制台。

IAM：访问策略模拟器控制台

此示例说明您如何创建基于身份的策略，以允许对附加到当前 Amazon Web Services 账户 中的用户、组或角色的策略使用策略模拟器控制台。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

您可以在以下位置访问 IAM policy simulator 控制台：https://policysim.aws.amazon.com/

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetGroup",
                "iam:GetGroupPolicy",
                "iam:GetPolicy",
                "iam:GetPolicyVersion",
                "iam:GetRole",
                "iam:GetRolePolicy",
                "iam:GetUser",
                "iam:GetUserPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListAttachedRolePolicies",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroups",
                "iam:ListGroupPolicies",
                "iam:ListGroupsForUser",
                "iam:ListRolePolicies",
                "iam:ListRoles",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "*"
        }
    ]
}
```

IAM：代入具有特定标签的角色

此示例说明了如何创建基于身份的策略以允许 IAM 用户代入具有标签键值对 `Project = ExampleCorpABC` 的角色。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

如果具有该标签的角色在与该用户相同的账户中存在，则用户可以担任该角色。如果具有该标签的角色在与该用户不同的账户中存在，则需要使用额外的权限。跨账户角色的信任策略还必须允许该用户或其账户的所有成员担任该角色。有关使用角色进行跨账户访问的信息，请参阅在您拥有的其他 Amazon Web Services 账户 中 IAM 用户的访问权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeTaggedRole",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:ResourceTag/Project": "ExampleCorpABC"}
            }
        }
    ]
}
```

IAM：允许或拒绝以编程方式和在控制台中访问多个服务

此示例说明了如何创建基于身份的策略，以允许完全访问多个服务并在 IAM 中进行有限的自我管理访问。该策略还会拒绝对 Amazon S3 `logs` 存储桶或 Amazon EC2 `i-1234567890abcdef0` 实例的访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

> **Warning**
>
> 该策略允许完全访问多个服务中的每个操作和资源。只应将该策略应用于受信任的管理员。

您可以将该策略作为权限边界，以定义基于身份的策略可以为 IAM 用户授予的最大权限。有关更多信息，请参阅 使用权限边界将责任委派给其他人。在将策略作为用户的权限边界时，这些语句定义以下边界：

- `AllowServices` 语句允许完全访问指定的 Amazon 服务。这意味着，用户在这些服务中的操作仅受附加到用户的权限策略的限制。
- `AllowIAMConsoleForCredentials` 语句允许访问以列出所有 IAM 用户。要在 Amazon Web Services 管理控制台中导航用户页面，此访问权限是必需的。它还允许查看账户的密码要求，这是用户更改自己的密码所必需的。
- `AllowManageOwnPasswordAndAccessKeys` 语句允许用户仅管理其自己的控制台密码和编程访问密钥。这是非常重要的，因为如果另一个策略为用户授予完全 IAM 访问权限，该用户可能会更改自己或其他用户的权限。此语句可防止这种情况的发生。
- `DenyS3Logs` 语句显式拒绝对 `logs` 存储桶的访问。该策略对用户实施公司限制。
- `DenyEC2Production` 语句显式拒绝对 `i-1234567890abcdef0` 实例的访问。

该策略不允许访问其他服务或操作。在将策略作为用户的权限边界时，即使附加到用户的其他策略允许这些操作，Amazon 也会拒绝请求。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowServices",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "cloudwatch:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowIAMConsoleForCredentials",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
```

```
            "Sid": "AllowManageOwnPasswordAndAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*LoginProfile*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::logs",
                "arn:aws:s3:::logs/*"
            ]
        },
        {
            "Sid": "DenyEC2Production",
            "Effect": "Deny",
            "Action": "ec2:*",
            "Resource": "arn:aws:ec2:*:*:instance/i-1234567890abcdef0"
        }
    ]
}
```

IAM：将特定标签添加到具有特定标签的用户

此示例说明了如何创建基于身份的策略以允许将带有标签值 Marketing、Development 或 QualityAssurance 的标签键 Department 添加到 IAM 用户。该用户必须已包含标签键值对 JobFunction = Manager。您可以使用此策略来要求一个经理仅属于三个部门之一。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

ListTagsForAllUsers 语句允许对您账户中的所有用户查看标签。

TagManagerWithSpecificDepartment 语句中的第一个条件使用 StringEquals 条件运算符。如果条件的两个部分都为 true，则此条件返回 true。要标记的用户必须已具有 JobFunction=Manager 标签。请求必须包括 Department 标签键以及其中一个列出的标签值。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中的标签键与策略中的键匹配，则条件返回 true。这意味着，请求中的唯一标签键必须为 Department。有关使用 ForAllValues 的更多信息，请参阅多值上下文键的集合运算符。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListTagsForAllUsers",
            "Effect": "Allow",
            "Action": [
                "iam:ListUserTags",
                "iam:ListUsers"
            ],
            "Resource": "*"
        },
        {
            "Sid": "TagManagerWithSpecificDepartment",
            "Effect": "Allow",
            "Action": "iam:TagUser",
            "Resource": "*",
            "Condition": {"StringEquals": {
                "iam:ResourceTag/JobFunction": "Manager",
                "aws:RequestTag/Department": [
                    "Marketing",
                    "Development",
                    "QualityAssurance"
                    ]
                },
                "ForAllValues:StringEquals": {"aws:TagKeys": "Department"}
            }
        }
    ]
}
```

IAM：添加具有特定值的特定标签

此示例说明了如何创建基于身份的策略以允许仅向任何 IAM 用户或角色添加标签键 CostCenter 和标签值 A-123 或标签值 B-456。您可以使用此策略将标记限制到一个特定的标签键或一组标签值。此策

略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

ConsoleDisplay 语句允许对您账户中的所有用户和角色查看标签。

AddTag 语句中的第一个条件使用 StringEquals 条件运算符。如果请求包含 CostCenter 标签键以及其中一个列出的标签值，则条件返回 true。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中的标签键与策略中的键匹配，则条件返回 true。这意味着，请求中的唯一标签键必须为 CostCenter。有关使用 ForAllValues 的更多信息，请参阅多值上下文键的集合运算符。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ConsoleDisplay",
            "Effect": "Allow",
            "Action": [
                "iam:GetRole",
                "iam:GetUser",
                "iam:ListRoles",
                "iam:ListRoleTags",
                "iam:ListUsers",
                "iam:ListUserTags"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AddTag",
            "Effect": "Allow",
            "Action": [
                "iam:TagUser",
                "iam:TagRole"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestTag/CostCenter": [
                        "A-123",
```

```
                    "B-456"
                ]
            },
            "ForAllValues:StringEquals": {"aws:TagKeys": "CostCenter"}
        }
    }
  ]
}
```

IAM：仅创建具有特定标签的新用户

此示例说明了如何创建基于身份的策略以允许创建 IAM 用户，但仅限于包含 Department 和 JobFunction 标签键之一或同时包含两者。Department 标签键必须具有 Development 或 QualityAssurance 标签值。JobFunction 标签键必须具有 Employee 标签值。您可以使用此策略要求新用户具有特定工作职能和部门。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*******替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

语句中的第一个条件使用 StringEqualsIfExists 条件运算符。如果请求中存在 Department 或 JobFunction 键，则标签必须包含指定的值。如果任一键都不存在，则此条件的计算结果将为 true。该条件的计算结果为 false 的唯一方法是：其中一个指定条件键在请求中存在，但具有的值不同于允许的值。有关使用 IfExists 的更多信息，请参阅...IfExists 条件运算符。

第二个条件使用 ForAllValues:StringEquals 条件运算符。如果请求中指定的每个标签键都与策略中至少一个值匹配，则该条件将返回 true。这意味着，请求中的所有标签必须在此列表中。但是，请求只能包含列表中的一个标签。例如，您可以创建一个仅具有 Department=QualityAssurance 标签的 IAM 用户。但是，您不能创建一个同时具有 JobFunction=Employee 标签和 Project=core 标签的 IAM 用户。有关使用 ForAllValues 的更多信息，请参阅多值上下文键的集合运算符。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "TagUsersWithOnlyTheseTags",
            "Effect": "Allow",
            "Action": [
                "iam:CreateUser",
```

```
                "iam:TagUser"
            ],
            "Resource": "*",
            "Condition": {
                "StringEqualsIfExists": {
                    "aws:RequestTag/Department": [
                        "Development",
                        "QualityAssurance"
                    ],
                    "aws:RequestTag/JobFunction": "Employee"
                },
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "Department",
                        "JobFunction"
                    ]
                }
            }
        }
    ]
}
```

IAM：生成和检索 IAM 凭证报告

此示例说明了如何创建基于身份的策略，以允许用户生成和下载报告，其中将列出其 Amazon Web Services 账户 中的所有 IAM 用户。该报告包含用户凭证的状态，包括密码、访问密钥、MFA 设备和签名证书。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

有关凭证报告的更多信息，请参阅为您的 Amazon Web Services 账户 生成凭证报告。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateCredentialReport",
            "iam:GetCredentialReport"
        ],
        "Resource": "*"
```

```
    }
}
```

IAM：允许以编程方式和在控制台中管理组的成员资格

此示例说明如何创建基于身份的策略以允许更新名为 MarketingTeam 组的成员资格。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的#######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

此策略有何作用？

- ViewGroups 语句允许用户列出 Amazon Web Services 管理控制台 中的所有用户和组。它还允许用户查看有关账户中的用户的基本信息。这些权限必须位于自己的语句中，因为它们不支持或不需要指定资源 ARN。相反，权限指定 "Resource" : "*"。
- ViewEditThisGroup 语句允许用户查看有关 MarketingTeam 组的信息，以及在该组中添加和删除用户。

该策略不允许用户查看或编辑用户或 MarketingTeam 组的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ViewGroups",
            "Effect": "Allow",
            "Action": [
                "iam:ListGroups",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:ListGroupsForUser"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ViewEditThisGroup",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:AddUserToGroup",
                "iam:RemoveUserFromGroup",
                "iam:GetGroup"
            ],
            "Resource": "arn:aws:iam::*:group/MarketingTeam"
        }
    ]
}
```

IAM：管理特定标签

此示例说明如何创建基于身份的策略以允许添加和删除来自 IAM 实体（用户和角色）且带有标签键 `Department` 的 IAM 标签。此策略不会限制 `Department` 标签的值。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:TagUser",
            "iam:TagRole",
            "iam:UntagUser",
            "iam:UntagRole"

        ],
        "Resource": "*",
        "Condition": {"ForAllValues:StringEquals": {"aws:TagKeys": "Department"}}
    }
}
```

IAM：将 IAM 角色传递给特定 Amazon 服务

此示例说明了如何创建基于身份的策略以允许将任何 IAM 服务角色传递给 Amazon CloudWatch 服务。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

服务角色是一个 IAM 角色，它将 Amazon 服务指定为可担任该角色的主体。这允许服务担任该角色，并代表您访问其他服务中的资源。要允许 Amazon CloudWatch 担任您传递的角色，必须在您的角色的信任策略中将 `cloudwatch.amazonaws.com` 服务主体指定为主体。服务主体由服务定义。要了解某项服务的服务主体，请参阅该服务的文档。对于某些服务，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色列为是的服务。请选择是与查看该服务的服务关联角色文档的链接。请搜索 `amazonaws.com` 以查看服务主体。

要了解将服务角色传递给服务的更多信息，请参阅向用户授予权限以将角色传递给 Amazon 服务。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:PassedToService":
 "cloudwatch.amazonaws.com"}
            }
        }
    ]
}
```

IAM：允许对 IAM 控制台进行只读访问而不生成报告

此示例说明了如何创建基于身份的策略以允许 IAM 用户执行任何以字符串 `Get` 或 `List` 开头的 IAM 操作。在用户使用控制台时，控制台向 IAM 发出请求以列出组、用户、角色和策略，并生成有关这些资源的报告。

星号将作为通配符。在策略中使用 `iam:Get*` 时，生成的权限包括以 `Get` 开头的所有 IAM 操作，例如，`GetUser` 和 `GetRole`。如果将来在 IAM 中添加新类型的实体，通配符是非常有用的。在这种情况下，策略授予的权限自动允许用户列出和获取有关这些新实体的详细信息。

此策略不能用于生成报告或服务的上次访问详细信息。有关允许此操作的其他策略，请参阅 IAM：允许对 IAM 控制台进行只读访问。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    }
}
```

IAM：允许对 IAM 控制台进行只读访问

此示例说明了如何创建基于身份的策略以允许 IAM 用户执行任何以字符串 `Get`、`List` 或 `Generate` 开头的 IAM 操作。在用户使用 IAM 控制台时，控制台发出请求以列出组、用户、角色和策略，并生成有关这些资源的报告。

星号将作为通配符。在策略中使用 `iam:Get*` 时，生成的权限包括以 `Get` 开头的所有 IAM 操作，例如，`GetUser` 和 `GetRole`。使用通配符是非常有用的，特别是将来在 IAM 中添加新类型的实体。在这种情况下，策略授予的权限自动允许用户列出和获取有关这些新实体的详细信息。

将此策略用于控制台访问，其中包括生成报告或服务上次访问详细信息的权限。有关不允许生成操作的其他策略，请参阅 IAM：允许对 IAM 控制台进行只读访问而不生成报告。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:Get*",
            "iam:List*",
            "iam:Generate*"
        ],
        "Resource": "*"
```

```
    }
}
```

IAM：允许特定的 IAM 用户以编程方式和在控制台中管理组

此示例说明如何创建基于身份的策略以允许特定 IAM 用户管理 `AllUsers` 组。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

此策略有何作用？

- `AllowAllUsersToListAllGroups` 语句允许列出所有组。这对于控制台访问是必需的。该权限必须位于自己的语句中，因为它不支持资源 ARN。相反，权限指定 `"Resource" : "*"`。
- `AllowAllUsersToViewAndManageThisGroup` 语句允许可以对组资源类型执行的所有组操作。它不允许执行 `ListGroupsForUser` 操作，可以对用户资源类型执行该操作，但不能对组资源类型执行该操作。有关可以为 IAM 操作指定的资源类型的更多信息，请参阅 Amazon Identity and Access Management 的操作、资源和条件键。
- `LimitGroupManagementAccessToSpecificUsers` 语句拒绝具有指定名称的用户访问以执行写入和权限管理组操作。如果策略中指定的用户尝试对组进行更改，该语句不会拒绝请求。`AllowAllUsersToViewAndManageThisGroup` 语句允许该请求。如果其他用户尝试执行这些操作，则会拒绝该请求。在 IAM 控制台中创建该策略时，您可以查看使用写入或权限管理访问级别定义的 IAM 操作。为此，请从 JSON 选项卡切换到可视化编辑器选项卡。有关访问级别的更多信息，请参阅 Amazon Identity and Access Management 的操作、资源和条件键。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAllUsersToListAllGroups",
            "Effect": "Allow",
            "Action": "iam:ListGroups",
            "Resource": "*"
        },
        {
            "Sid": "AllowAllUsersToViewAndManageThisGroup",
            "Effect": "Allow",
```

```
            "Action": "iam:*Group*",
            "Resource": "arn:aws:iam::*:group/AllUsers"
        },
        {
            "Sid": "LimitGroupManagementAccessToSpecificUsers",
            "Effect": "Deny",
            "Action": [
                "iam:AddUserToGroup",
                "iam:CreateGroup",
                "iam:RemoveUserFromGroup",
                "iam:DeleteGroup",
                "iam:AttachGroupPolicy",
                "iam:UpdateGroup",
                "iam:DetachGroupPolicy",
                "iam:DeleteGroupPolicy",
                "iam:PutGroupPolicy"
            ],
            "Resource": "arn:aws:iam::*:group/AllUsers",
            "Condition": {
                "StringNotEquals": {
                    "aws:username": [
                        "srodriguez",
                        "mjackson",
                        "adesai"
                    ]
                }
            }
        }
    ]
}
```

IAM：允许以编程方式和在控制台中设置账户密码要求

此示例说明了如何创建基于身份的策略以允许用户查看和更新其账户密码要求。密码要求指定账户成员密码的复杂性要求和强制轮换期。此策略定义了程序访问和控制台访问的权限。

要了解如何为您的账户设置账户密码要求策略，请参阅为 IAM 用户设置账户密码策略。

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GetAccountPasswordPolicy",
            "iam:UpdateAccountPasswordPolicy"
        ],
        "Resource": "*"
    }
}
```

IAM：基于用户路径访问策略模拟器 API

此示例说明了如何创建基于身份的策略以仅允许具有路径 Department/Development 的用户使用策略模拟器 API。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetContextKeysForPrincipalPolicy",
                "iam:SimulatePrincipalPolicy"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

> **Note**
>
> 要创建允许具有路径 Department/Development 的用户使用策略模拟器控制台的策略，请参阅 IAM：基于用户路径访问策略模拟器控制台。

IAM：基于用户路径访问策略模拟器控制台

此示例说明了如何创建基于身份的策略以仅允许具有路径 Department/Development 的用户使用策略模拟器控制台。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

您可以在以下位置访问 IAM policy simulator：https://policysim.aws.amazon.com/

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "iam:GetPolicy",
                "iam:GetUserPolicy"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "iam:GetUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListGroupsForUser",
                "iam:ListUserPolicies",
                "iam:ListUsers"
            ],
            "Effect": "Allow",
            "Resource": "arn:aws:iam::*:user/Department/Development/*"
        }
    ]
}
```

IAM：允许 IAM 用户自行管理 MFA 设备

此示例说明如何创建 IAM policy 以允许基于身份的用户自行管理其多重身份验证（MFA）设备。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

> **Note**
>
> 如果具有此策略的 IAM 用户未通过 MFA 进行身份验证，则此策略会拒绝对所有 Amazon 操作（使用 MFA 进行身份验证所需的操作除外）的访问。如果为已登录 Amazon 的用户添加这些权限，他们需要注销并重新登录才能看到这些更改。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowListActions",
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:ListVirtualMFADevices"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUserToCreateVirtualMFADevice",
            "Effect": "Allow",
            "Action": [
                "iam:CreateVirtualMFADevice"
            ],
            "Resource": "arn:aws:iam::*:mfa/*"
        },
        {
            "Sid": "AllowUserToManageTheirOwnMFA",
            "Effect": "Allow",
            "Action": [
                "iam:EnableMFADevice",
                "iam:GetMFADevice",
                "iam:ListMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        },
        {
            "Sid": "AllowUserToDeactivateTheirOwnMFAOnlyWhenUsingMFA",
```

```
            "Effect": "Allow",
            "Action": [
                "iam:DeactivateMFADevice"
            ],
            "Resource": [
                "arn:aws:iam::*:user/${aws:username}"
            ],
            "Condition": {
                "Bool": {
                    "aws:MultiFactorAuthPresent": "true"
                }
            }
        },
        {
            "Sid": "BlockMostAccessUnlessSignedInWithMFA",
            "Effect": "Deny",
            "NotAction": [
                "iam:CreateVirtualMFADevice",
                "iam:EnableMFADevice",
                "iam:ListMFADevices",
                "iam:ListUsers",
                "iam:ListVirtualMFADevices",
                "iam:ResyncMFADevice"
            ],
            "Resource": "*",
            "Condition": {
                "BoolIfExists": {
                    "aws:MultiFactorAuthPresent": "false"
                }
            }
        }
    ]
}
```

IAM：允许 IAM 用户通过编程方式和控制台更新自己的凭证

此示例演示了如何创建基于身份的策略以允许 IAM 用户更新自己的访问密钥、签名证书、特定于服务的凭证和密码。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:ListUsers",
                "iam:GetAccountPasswordPolicy"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:*AccessKey*",
                "iam:ChangePassword",
                "iam:GetUser",
                "iam:*ServiceSpecificCredential*",
                "iam:*SigningCertificate*"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        }
    ]
}
```

要了解用户如何在控制台中更改自己的密码，请参阅the section called “IAM 用户如何更改自己的密码”。

IAM：查看 Amazon Organizations 策略的服务上次访问信息

此示例说明了如何创建基于身份的策略以允许查看特定 Amazon Organizations 策略的服务上次访问信息。该策略允许检索具有 p-policy123 ID 的服务控制策略 (SCP) 的数据。必须使用 Amazon Organizations 管理账户凭证对生成和查看报告的人员进行身份验证。该策略允许请求者检索其组织中的任何 Amazon Organizations 实体的数据。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的#######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

有关上次访问的信息的重要信息（包括所需的权限、故障排除和支持的区域），请参阅使用上次访问的信息优化 Amazon 中的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowOrgsReadOnlyAndIamGetReport",
            "Effect": "Allow",
            "Action": [
                "iam:GetOrganizationsAccessReport",
                "organizations:Describe*",
                "organizations:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowGenerateReportOnlyForThePolicy",
            "Effect": "Allow",
            "Action": "iam:GenerateOrganizationsAccessReport",
            "Resource": "*",
            "Condition": {
                "StringEquals": {"iam:OrganizationsPolicyId": "p-policy123"}
            }
        }
    ]
}
```

IAM：对可应用于 IAM 用户、组或角色的托管策略加以限制

此示例说明了如何创建基于身份的策略以限制客户托管和 Amazon 托管的策略，这些策略将应用于 IAM 用户、组或角色。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的#######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:AttachUserPolicy",
            "iam:DetachUserPolicy"
        ],
        "Resource": "*",
        "Condition": {
            "ArnEquals": {
                "iam:PolicyARN": [
                    "arn:aws:iam::*:policy/policy-name-1",
                    "arn:aws:iam::*:policy/policy-name-2"
                ]
            }
        }
    }
}
```

Amazon Lambda：允许 Lambda 函数访问 Amazon DynamoDB 表

此示例说明了如何创建基于身份的策略以允许对特定 Amazon DynamoDB 表进行读写访问。该策略还允许将日志文件写入到 CloudWatch Logs 中。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

要使用该策略，请将策略附加到 Lambda 服务角色。服务角色是在您的账户中创建的一个角色，以允许服务代表您执行操作。该服务角色必须在信任策略中包含 Amazon Lambda 以作为主体。有关如何使用此策略的详细信息，请参阅 Amazon 安全博客中的如何创建 Amazon IAM policy 以授予 Amazon Lambda 访问 Amazon DynamoDB 表的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ReadWriteTable",
            "Effect": "Allow",
            "Action": [
                "dynamodb:BatchGetItem",
                "dynamodb:GetItem",
```

```
                "dynamodb:Query",
                "dynamodb:Scan",
                "dynamodb:BatchWriteItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable"
        },
        {
            "Sid": "GetStreamRecords",
            "Effect": "Allow",
            "Action": "dynamodb:GetRecords",
            "Resource": "arn:aws:dynamodb:*:*:table/SampleTable/stream/* "
        },
        {
            "Sid": "WriteLogStreamsAndGroups",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogStream",
                "logs:PutLogEvents"
            ],
            "Resource": "*"
        },
        {
            "Sid": "CreateLogGroup",
            "Effect": "Allow",
            "Action": "logs:CreateLogGroup",
            "Resource": "*"
        }
    ]
}
```

Amazon RDS：允许在特定区域中进行完全 RDS 数据库访问。

此示例说明了如何创建基于身份的策略以允许在特定区域中进行完全 RDS 数据库访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的######替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "rds:*",
            "Resource": ["arn:aws:rds:us-east-1:*:*"]
        },
        {
            "Effect": "Allow",
            "Action": ["rds:Describe*"],
            "Resource": ["*"]
        }
    ]
}
```

Amazon RDS：允许以编程方式和在控制台中还原 RDS 数据库

此示例说明了了如何创建允许还原 RDS 数据库的基于身份的策略。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "rds:CreateDBParameterGroup",
                "rds:CreateDBSnapshot",
                "rds:DeleteDBSnapshot",
                "rds:Describe*",
                "rds:DownloadDBLogFilePortion",
                "rds:List*",
                "rds:ModifyDBInstance",
                "rds:ModifyDBParameterGroup",
                "rds:ModifyOptionGroup",
                "rds:RebootDBInstance",
                "rds:RestoreDBInstanceFromDBSnapshot",
                "rds:RestoreDBInstanceToPointInTime"
```

```
            ],
            "Resource": "*"
        }
    ]
}
```

Amazon RDS：允许标签所有者不受限制地访问其标记的 RDS 资源

此示例显示您可以创建一个基于身份的策略，对标签所有者授予对已标记的 RDS 资源的完全访问权限。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Action": [
                "rds:Describe*",
                "rds:List*"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": [
                "rds:DeleteDBInstance",
                "rds:RebootDBInstance",
                "rds:ModifyDBInstance"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:db-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyOptionGroup",
                "rds:DeleteOptionGroup"
            ],
```

```
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:og-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyDBParameterGroup",
                "rds:ResetDBParameterGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:pg-tag/Owner": "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:AuthorizeDBSecurityGroupIngress",
                "rds:RevokeDBSecurityGroupIngress",
                "rds:DeleteDBSecurityGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:secgrp-tag/Owner":
"${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:DeleteDBSnapshot",
                "rds:RestoreDBInstanceFromDBSnapshot"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:snapshot-tag/Owner":
"${aws:username}"}
            }
        },
        {
            "Action": [
```

```
                "rds:ModifyDBSubnetGroup",
                "rds:DeleteDBSubnetGroup"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:subgrp-tag/Owner":
 "${aws:username}"}
            }
        },
        {
            "Action": [
                "rds:ModifyEventSubscription",
                "rds:AddSourceIdentifierToSubscription",
                "rds:RemoveSourceIdentifierFromSubscription",
                "rds:DeleteEventSubscription"
            ],
            "Effect": "Allow",
            "Resource": "*",
            "Condition": {
                "StringEqualsIgnoreCase": {"rds:es-tag/Owner": "${aws:username}"}
            }
        }
    ]
}
```

Amazon S3：允许 Amazon Cognito 用户访问其存储桶中的对象

此示例说明了如何创建基于身份的策略以允许 Amazon Cognito 用户访问特定 Amazon S3 存储桶中的对象。该策略仅允许访问名称包含 `cognito`、应用程序名称以及联合主体 ID 的对象，由 ${cognito-identity.amazonaws.com:sub} 变量表示。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

> **Note**
>
> 对象键中使用的“sub”值不是用户池中的 sub 值，而是与身份池中的用户关联的身份 ID。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ListYourObjects",
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": [
        "arn:aws:s3:::bucket-name"
      ],
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "cognito/application-name/${cognito-identity.amazonaws.com:sub}/*"
          ]
        }
      }
    },
    {
      "Sid": "ReadWriteDeleteYourObjects",
      "Effect": "Allow",
      "Action": [
        "s3:DeleteObject",
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::bucket-name/cognito/application-name/${cognito-
identity.amazonaws.com:sub}/*"
      ]
    }
  ]
}
```

Amazon Cognito 为您的 Web 和移动应用程序提供身份验证、授权和用户管理。您的用户可使用用户名和密码直接登录，也可以通过第三方 (如 Facebook、Amazon 或 Google) 登录。

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的应用程序提供注册和登录选项的用户目录。使用身份池，您可以授予用户访问其他 Amazon 服务的权限。您可以单独或配合使用身份池和用户池。

有关 Amazon Cognito 的更多信息，请参阅 Amazon Cognito 用户指南。

Amazon S3：允许联合用户以编程方式和在控制台中访问其 Amazon S3 主目录

此示例说明了如何创建基于身份的策略以允许联合主体访问其位于 S3 中的主目录存储桶对象。主目录是一个包含 home 文件夹和个人联合主体文件夹的存储桶。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

此策略中的 `${aws:userid}` 变量解析为 `role-id:specified-name`。联合主体 ID 的 `role-id` 部分是创建过程中分配给联合主体角色的唯一标识符。有关更多信息，请参阅 唯一标识符。`specified-name` 是当联合主体担任其角色时传递给 `AssumeRoleWithWebIdentity` 请求的 RoleSessionName 参数。

您可使用 Amazon CLI 命令 `aws iam get-role --role-name` *`specified-name`* 查看此角色 ID。例如，假设您指定友好名称 `John` 并且 CLI 返回角色 ID `AROAXXT2NJT7D3SIQN7Z6`。在此示例中，联合主体的用户 ID 为 `AROAXXT2NJT7D3SIQN7Z6:John`。此策略之后允许联合主体 John 访问具有前缀 `AROAXXT2NJT7D3SIQN7Z6:John` 的 Amazon S3 存储桶。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
```

```
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
                        "",
                        "home/",
                        "home/${aws:userid}/*"
                    ]
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:userid}",
                "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:userid}/*"
            ]
        }
    ]
}
```

Amazon S3：S3 存储桶访问，但在最近未进行 MFA 的情况下拒绝生产存储桶

此示例说明如何创建基于身份的策略以允许 Amazon S3 管理员访问任何存储桶（包括更新、添加和删除对象）。不过，如果用户在过去 30 分钟内未使用多重身份验证 (MFA)??? 登录，则显式拒绝访问 `amzn-s3-demo-bucket-production` 存储桶。该策略授予所需的权限以在控制台中执行该操作，或者以编程方式使用 Amazon CLI 或 Amazon API 执行该操作。要使用此策略，请将示例策略中的*########*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

该策略从不允许使用长期用户访问密钥以编程方式访问 `amzn-s3-demo-bucket` 存储桶。这是将 `aws:MultiFactorAuthAge` 条件键与 `NumericGreaterThanIfExists` 条件运算符结合使用来实现的。如果 MFA 不存在或 MFA 的使用期限超过 30 分钟，则此策略条件将返回 `true`。在这些情况下，访问将被拒绝。要以编程方式访问 `amzn-s3-demo-bucket-production` 存储桶，S3 管理员必须使用通过 GetSessionToken API 操作在过去 30 分钟内生成的临时凭证。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListAllS3Buckets",
            "Effect": "Allow",
            "Action": ["s3:ListAllMyBuckets"],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketLevelActions",
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket",
                "s3:GetBucketLocation"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowBucketObjectActions",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl",
                "s3:GetObject",
                "s3:GetObjectAcl",
                "s3:DeleteObject"
            ],
            "Resource": "arn:aws:s3:::*/*"
        },
        {
            "Sid": "RequireMFAForProductionBucket",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-production/*",
                "arn:aws:s3:::amzn-s3-demo-bucket-production"
            ],
            "Condition": {
                "NumericGreaterThanIfExists": {"aws:MultiFactorAuthAge": "1800"}
            }
```

```
        }
    ]
}
```

Amazon S3：允许 IAM 用户以编程方式和在控制台中访问其 S3 主目录

此示例说明了如何创建基于身份的策略以允许 IAM 用户访问其位于 S3 中的主目录存储桶对象。主目录是一个包含 home 文件夹和个人用户文件夹的存储桶。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

使用 IAM 角色时，此策略将不起作用，因为使用角色 IAM 时 `aws:username` 变量不可用。有关主体键值的详细信息，请参阅 主体键值。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Condition": {
                "StringLike": {
                    "s3:prefix": [
```

```
                    "",
                    "home/",
                    "home/${aws:username}/*"
                ]
            }
        }
    },
    {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": [
            "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}",
            "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}/*"
        ]
    }
  ]
}
```

Amazon S3：将管理限制为特定 S3 存储桶

此示例说明了如何创建基于身份的策略以限制 Amazon S3 存储桶对该特定存储桶的管理。此策略授予执行所有 Amazon S3 操作的权限，但拒绝访问除 Amazon S3 外的所有 Amazon Web Services 服务。请参阅以下示例。根据此策略，您只能访问可以对 S3 存储桶或 S3 对象资源执行的 Amazon S3 操作。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

如果将该策略与允许该策略拒绝的操作的其他策略（例如 AmazonS3FullAccess 或 AmazonEC2FullAccess Amazon 托管策略）一起使用，则访问会被拒绝。这是因为“显式拒绝”声明优先于“允许”声明。有关更多信息，请参阅 Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。

> **Warning**
>
> `NotAction` 和 `NotResource` 是必须谨慎使用的高级策略元素。该策略将拒绝对除 Amazon S3 以外所有 Amazon 服务的访问。如果将该策略挂载到用户，则授予其他服务访问权限的任何其他策略都会被忽略 (访问会被拒绝)。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        },
        {
            "Effect": "Deny",
            "NotAction": "s3:*",
            "NotResource": [
                "arn:aws:s3:::bucket-name",
                "arn:aws:s3:::bucket-name/*"
            ]
        }
    ]
}
```

授予对 Amazon S3 存储桶对象的读取和写入访问权限

此示例说明如何创建基于身份的策略以允许 `Read` 和 `Write` 对特定 Amazon S3 存储桶中的对象进行访问。此策略授予有计划地通过 Amazon API 或 Amazon CLI 完成此操作的必要权限。要使用此策略，请将示例策略中的`#######`替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

`s3:*Object` 操作使用通配符作为操作名称的一部分。`AllObjectActions` 语句允许以单词“Object”结尾的 `GetObject`、`DeleteObject`、`PutObject` 和任何其他 Amazon S3 操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": ["s3:ListBucket"],
            "Resource": ["arn:aws:s3:::bucket-name"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::bucket-name/*"]
        }
    ]
}
```

> **Note**
>
> 要允许对 Amazon S3 存储桶中的对象进行 Read 和 Write 访问并包含用于控制台访问的其他权限，请参阅 Amazon S3：允许以编程方式和在控制台中对 S3 存储桶中的对象进行读写访问。

Amazon S3：允许以编程方式和在控制台中对 S3 存储桶中的对象进行读写访问

此示例说明如何创建基于身份的策略以允许 Read 和 Write 对特定 S3 存储桶中的对象进行访问。此策略定义了程序访问和控制台访问的权限。要使用此策略，请将示例策略中的*******替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

s3:*Object 操作使用通配符作为操作名称的一部分。AllObjectActions 语句允许以单词“Object”结尾的 GetObject、DeleteObject、PutObject 和任何其他 Amazon S3 操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "S3ConsoleAccess",
            "Effect": "Allow",
            "Action": [
                "s3:GetAccountPublicAccessBlock",
```

```
                "s3:GetBucketAcl",
                "s3:GetBucketLocation",
                "s3:GetBucketPolicyStatus",
                "s3:GetBucketPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "*"
        },
        {
            "Sid": "ListObjectsInBucket",
            "Effect": "Allow",
            "Action": "s3:ListBucket",
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"]
        },
        {
            "Sid": "AllObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object",
            "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/*"]
        }
    ]
}
```

管理 IAM 策略

IAM 为您提供了相应的工具以创建和管理所有类型的 IAM policy（托管策略和内联策略）。要向 IAM 身份（IAM 用户、组或角色）添加权限，请创建策略、验证策略，然后将该策略附加到身份。您可以将多个策略附加到一个身份，每个策略可以包含多个权限。

主题

- 其他资源
- 使用客户管理型策略定义自定义 IAM 权限
- IAM 策略验证
- 使用 IAM policy simulator 测试 IAM 策略
- 添加和删除 IAM 身份权限
- IAM 策略版本控制
- 编辑 IAM 策略

- 删除 IAM 策略
- 使用上次访问的信息优化 Amazon 中的权限

其他资源

以下资源可帮助您了解有关 Amazon 的更多信息。

- 有关不同类型的 IAM policy 的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限。
- 有关使用 IAM 中的策略的一般信息，请参阅 适用于 Amazon 资源的 Access Management。
- 有关如何使用 IAM Acess Analyzer 生成基于实体的访问活动的 IAM 策略的信息，请参阅 IAM Access Analyzer 策略生成。
- 有关在多个策略对给定 IAM 身份有效时如何评估权限的信息，请参阅 策略评估逻辑。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

使用客户管理型策略定义自定义 IAM 权限

策略定义 Amazon 中的身份或资源的权限。您可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 在 IAM 中创建客户管理型策略。客户管理型策略是您在自己的 Amazon Web Services 账户 中管理的独立策略。然后，您可以将策略附加到您的 Amazon Web Services 账户 中的身份（用户、组和角色）。

基于身份的策略是附加到 IAM 中身份的策略。基于身份的策略可以包括 Amazon 托管策略、客户管理型策略和内联策略。Amazon 托管策略由 Amazon 创建和管理，您可以使用但无法管理这些策略。内联策略是您创建并直接嵌入到 IAM 用户组、用户或角色的策略。内联策略不能在其他身份上重复使用，也不能在其所存在的身份之外托管。有关更多信息，请参阅 添加和删除 IAM 身份权限。

通常来说，最好使用客户管理型策略而非内联策略或 Amazon 托管策略。Amazon 托管策略通常提供广泛的管理权限或只读权限。为了获得最高安全性，请授予最低权限，这意味着仅授予执行特定作业任务所需的权限。

创建或编辑 IAM policy 时，Amazon 可以自动执行策略验证，以帮助您创建具有最低权限的有效策略。在 Amazon Web Services 管理控制台 中，IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

您可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 在 IAM 中创建客户托管策略。有关使用 Amazon CloudFormation 模板添加或更新策略的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 Amazon Identity and Access Management 资源类型参考。

主题

- 创建 IAM 策略（控制台）
- 创建 IAM 策略（Amazon CLI）
- 创建 IAM 策略（Amazon API）

创建 IAM 策略（控制台）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon Web Services 管理控制台 在 IAM 中创建客户托管策略。客户管理型策略是您在自己的 Amazon Web Services 账户 中管理的独立策略。然后，您可以将策略附加到您的 Amazon Web Services 账户 中的身份（用户、组和角色）。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

主题

- 创建 IAM 策略
- 使用 JSON 编辑器创建策略
- 使用可视化编辑器创建策略
- 导入现有托管策略

创建 IAM 策略

您可以使用下列方法之一在 Amazon Web Services 管理控制台中创建客户托管策略：

- JSON - 粘贴和自定义已发布的基于身份的示例策略。
- Visual editor（可视化编辑器）- 在可视化编辑器中从头开始构建新的策略。如果使用可视化编辑器，您不需要了解 JSON 语法。
- Import（导入）- 从您的账户中导入并自定义托管策略。您可以导入之前创建的 Amazon 托管策略或客户托管策略。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

使用 JSON 编辑器创建策略

您可以选择 JSON 选项以在 JSON 中键入或粘贴策略。这种方法适用于复制示例策略以在您的账户中使用。或者，您可以在 JSON 编辑器中键入自己的 JSON 策略文档。也可以使用 JSON 选项在可视化编辑器和 JSON 之间切换以比较这些视图。

当您在 JSON 编辑器中创建或编辑策略时，IAM 会执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 提供额外的策略检查和可执行的建议，以帮助您进一步优化策略。

JSON 策略文档包含一个或多个语句。每个语句应包含具有相同效果 (`Allow` 或 `Deny`) 并支持相同资源和条件的所有操作。如果一个操作要求指定所有资源 ("*")，而另一个操作支持特定资源的 Amazon Resource Name (ARN)，则它们必须位于两个单独的 JSON 语句中。有关 ARN 格式的详细信息，请参阅《Amazon Web Services 一般参考指南》中的 Amazon 资源名称 (ARN) 。有关 IAM policy 的一般信息，请参阅 Amazon Identity and Access Management 中的策略和权限。有关 IAM policy 语言的信息，请参阅 IAM JSON 策略参考。

使用 JSON 策略编辑器创建策略

1. 登录Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项。
5. 键入或粘贴一个 JSON 策略文档。有关 IAM policy 语言的详细信息，请参阅 IAM JSON 策略参考。
6. 解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

7. （可选）在 Amazon Web Services 管理控制台 中创建或编辑策略时，您可以生成可在 Amazon CloudFormation 模板中使用的 JSON 或 YAML 策略模板。

   为此，请在策略编辑器中选择操作，然后选择生成 CloudFormation 模板。如需了解有关 Amazon CloudFormation 的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 Amazon Identity and Access Management资源类型参考。
8. 向策略添加完权限后，选择下一步。
9. 在查看和创建页面上，为创建的策略键入策略名称和描述（可选）。查看此策略中定义的权限以查看您的策略授予的权限。
10. （可选）通过以密钥值对的形式附加标签来向策略添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
11. 选择创建策略可保存您的新策略。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅 添加和删除 IAM 身份权限。

使用可视化编辑器创建策略

IAM 控制台中的可视化编辑器指导您完成创建策略的过程，而无需编写 JSON 语法。要查看使用可视化编辑器创建策略的示例，请参阅the section called “控制对身份的访问”。

使用可视化编辑器创建策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。
4. 在策略编辑器部分中，找到选择服务部分，然后选择 Amazon 服务。您可以使用顶部的搜索框限制服务列表中的结果。您只能在一个可视化编辑器权限块中选择一个服务。要为多个服务授予访问权限，请选择添加更多权限以添加多个权限块。
5. 对于允许的操作，选择要添加到策略的操作。您可以使用以下方法选择操作：

   - 选中所有操作的复选框。
   - 选择添加操作以键入特定操作的名称。您可以使用通配符 (*) 指定多个操作。
   - 选择访问级别组之一以选择访问级别的所有操作（例如，读取、写入或列出）。
   - 展开每个访问级别组以选择单独的操作。

预设情况下，您创建的策略允许执行选择的操作。要拒绝选择的操作，请选择切换到拒绝权限。由于 IAM 默认拒绝，作为安全最佳实践，我们建议您仅允许用户所需的操作和资源的权限。只有在要覆盖由其他语句或策略单独允许的权限时，您才应创建 JSON 语句以拒绝权限。我们建议您将拒绝权限数限制为最低，因为它们可能会增加解决权限问题的难度。

6. 对于资源，如果您在前面步骤中选择的服务和操作不支持选择特定资源，则允许使用所有资源，并且您无法编辑此部分。

   如果您选择了一个或多个操作支持资源级权限，可视化编辑器会列出这些资源。然后，展开资源以指定您的策略的资源。

   您可以通过以下方式指定资源：

   - 选择添加 ARN，通过 Amazon 资源名称（ARN）指定资源。您可以使用可视化 ARN 编辑器或手动列出 ARN。有关 ARN 语法的更多信息，请参阅《Amazon Web Services 一般参考指南》中的 Amazon 资源名称（ARN）。有关在策略的 `Resource` 元素中使用 ARN 的信息，请参阅IAM JSON 策略元素：Resource。
   - 选择资源旁边的此账户中的任意项以授予对该类型的任何资源的权限。
   - 选择所有以选择该服务的所有资源。

7. （可选）选择请求条件 – 可选，以在创建的策略中添加条件。条件限制 JSON 策略语句的效果。例如，您可以指定仅在以下情况下允许用户对资源执行操作：该用户的请求发生在特定的时间范围内。您还可以使用常用的条件来限制是否必须使用多重验证 (MFA) 设备来验证用户身份。或者，您可以要求请求来自特定范围的 IP 地址。有关您可以在策略条件中使用的所有上下文键的列表，请参阅服务授权参考中的 Amazon 服务的操作、资源和条件键。

   您可以使用以下方法选择条件：

   - 使用复选框选择常用的条件。
   - 选择添加其他条件以指定其他条件。选择条件的条件键、限定词和运算符，然后键入一个值。要添加多个值，请选择添加。您可以将这些值视为通过逻辑“OR”运算符连接在一起。完成后，选择添加条件。

   要添加多个条件，请再次选择添加其他条件。根据需要重复上述步骤。每个条件仅适用于该可视化编辑器权限块。所有条件都必须为 true，才会将权限块视为匹配项。也就是说，将条件视为通过逻辑“AND”运算符连接在一起。

有关 Condition 元素的更多信息，请参阅 IAM JSON 策略元素：Condition中的 IAM JSON 策略参考。

8. 要添加更多权限块，请选择添加更多权限。对于每个块，重复步骤 2 到步骤 5。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

9. （可选）在 Amazon Web Services 管理控制台 中创建或编辑策略时，您可以生成可在 Amazon CloudFormation 模板中使用的 JSON 或 YAML 策略模板。

   为此，请在策略编辑器中选择操作，然后选择生成 CloudFormation 模板。如需了解有关 Amazon CloudFormation 的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 Amazon Identity and Access Management资源类型参考。

10. 向策略添加完权限后，选择下一步。
11. 在查看和创建页面上，为创建的策略键入策略名称和描述（可选）。查看此策略中定义的权限，确保您授予了所需的权限。
12. （可选）通过以密钥值对的形式附加标签来向策略添加元数据。有关在 IAM 中使用标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
13. 选择创建策略可保存您的新策略。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅 添加和删除 IAM 身份权限。

导入现有托管策略

要创建新的策略，一种简单的方法是在您的账户中导入至少具有一部分所需权限的现有托管策略。您随后可以自定义该策略，使其符合您的新要求。

您无法导入内联策略。要了解托管策略和内联策略之间的差别，请参阅托管策略与内联策略。

在可视化编辑器中导入现有的托管策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。

2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。
4. 在策略编辑器中，选择可视化，然后在页面右侧选择操作，然后选择导入策略。
5. 在导入策略窗口中，选择与要在新策略中包含的策略最匹配的管理型策略。您可以使用顶部的搜索框限制策略列表中的结果。
6. 选择导入策略。

   将在策略底部的新权限块中添加导入的策略。
7. 使用可视化编辑器或选择 JSON 以自定义您的策略。然后选择下一步。

   > **Note**
   >
   > 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。
8. 在查看和创建页面上，为创建的策略键入策略名称和描述（可选）。您以后无法编辑这些设置。查看此策略中定义的权限，然后选择创建策略以保存您的工作。

在 JSON 编辑器中导入现有的管理型策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。
3. 选择创建策略。
4. 在策略编辑器部分，选择 JSON 选项，然后在页面右侧选择操作，然后选择导入策略。
5. 在导入策略窗口中，选择与要在新策略中包含的策略最匹配的管理型策略。您可以使用顶部的搜索框限制策略列表中的结果。
6. 选择导入策略。

   导入的策略中的语句将添加到 JSON 策略底部。
7. 在 JSON 中自定义您的策略。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。或者，在 Visual editor（可视化编辑器）中自定义您的策略。然后选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

8. 在查看和创建页面上，为创建的策略键入策略名称和描述（可选）。您以后无法编辑这些字段。查看策略此策略中定义的权限，然后选择创建策略以保存您的工作。

在创建策略后，可以将其附加到您的组、用户或角色。有关更多信息，请参阅 添加和删除 IAM 身份权限。

创建 IAM 策略（Amazon CLI）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon CLI 在 IAM 中创建客户托管策略。客户管理型策略是您在自己的 Amazon Web Services 账户 中管理的独立策略。作为最佳实践，我们建议您使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性。通过验证策略，您可以在将策略附加到您 Amazon Web Services 账户 中的身份（用户、组和角色）之前，解决任何错误或建议。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

创建 IAM policy (Amazon CLI)

您可以使用 Amazon Command Line Interface (Amazon CLI) 创建 IAM 客户托管策略或内联策略。

创建客户托管策略 (Amazon CLI)

使用以下命令：

- create-policy

为 IAM 身份（组、用户或角色）创建内联策略 (Amazon CLI)

使用以下命令之一：

- put-group-policy
- put-role-policy

- put-user-policy

> **Note**
>
> 您不能使用 IAM 嵌入服务相关角色的内联策略。

验证客户托管策略 (Amazon CLI)

使用以下 IAM Access Analyzer 命令：

- validate-policy

创建 IAM 策略（Amazon API）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon API 在 IAM 中创建客户托管策略。客户管理型策略是您在自己的 Amazon Web Services 账户 中管理的独立策略。作为最佳实践，我们建议您使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性。通过验证策略，您可以在将策略附加到您 Amazon Web Services 账户 中的身份（用户、组和角色）之前，解决任何错误或建议。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

创建 IAM policy (Amazon API)

您可以使用 Amazon API 创建 IAM 客户托管策略或内联策略。

创建客户托管策略 (Amazon API)

调用以下操作：

- CreatePolicy

为 IAM 身份（组、用户或角色）创建内联策略 (Amazon API)

调用下列一项操作：

- PutGroupPolicy
- PutRolePolicy

- PutUserPolicy

> **Note**
>
> 您不能使用 IAM 嵌入服务相关角色的内联策略。

验证客户托管策略 (Amazon API)

调用以下 IAM Access Analyzer 操作：

- ValidatePolicy

IAM 策略验证

策略是使用 IAM policy 语法编写的 JSON 文档。当您将策略附加到 IAM 实体（例如用户、组或角色）时，策略将向该实体授予权限。

当您使用 Amazon Web Services 管理控制台 创建或编辑 IAM 访问控制策略时，Amazon 会自动检查它们，以确保它们符合 IAM policy 语法。如果 Amazon 确定策略不符合语法，则会提示您修复策略。

IAM Access Analyzer 提供额外的策略检查和建议，以帮助您进一步优化策略。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。要查看 IAM Access Analyzer 返回的警告、错误和建议的列表，请参阅 IAM Access Analyzer 策略检查参考。

验证范围

Amazon 检查 JSON 策略语法。它还验证您的 ARN 格式是否正确，操作名称和条件键是否正确。

访问策略验证

当您创建 JSON 策略或在 Amazon Web Services 管理控制台 中编辑现有策略时，将自动对策略进行验证。如果策略语法无效，您会收到通知，必须先解决问题，然后才能继续。如果您有适用于 `access-analyzer:ValidatePolicy` 的权限，则 IAM Access Analyzer 策略验证的结果会自动返回到 Amazon Web Services 管理控制台。您还可以使用 Amazon API 或 Amazon CLI 验证策略。

现有策略

您的现有策略可能无效，因为它们是在策略引擎的最新更新之前创建或保存的。作为最佳实践，我们建议您使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性。我们建议您打开现有策略并查看生成的策略验证结果。如果未修复任何策略语法错误，则无法编辑和保存现有策略。

使用 IAM policy simulator 测试 IAM 策略

有关如何以及为何使用 IAM policy 的更多信息，请参阅。Amazon Identity and Access Management 中的策略和权限

您可以在以下位置访问 IAM policy simulator 控制台：https://policysim.aws.amazon.com/

> **Important**
>
> 策略模拟器的结果可能与您的实时 Amazon 环境不同。我们建议您在使用策略模拟器进行测试后，根据真实 Amazon 环境检查您的策略，以确认您获得了预期的结果。有关更多信息，请参阅 IAM policy simulator 的工作方式。

IAM policy simulator 入门

使用 IAM policy simulator，您可以测试和排除基于身份的策略和 IAM 权限边界的故障。以下是您可以使用策略模拟器执行的一些常见操作：

- 测试附加到您 Amazon Web Services 账户 中的 IAM 用户、IAM 组或角色的基于身份的策略。如果多个策略附加到用户、用户组或角色，您可以测试所有策略，也可以选择单个策略进行测试。您可以测试为特定资源所选的策略允许或拒绝哪些操作。
- 测试权限边界对 IAM 实体的影响并进行故障排除。一次只能模拟一个权限边界。
- 测试基于资源的策略对附加到 Amazon 资源（例如 Amazon S3 存储桶、Amazon SQS 队列、Amazon SNS 主题或 Amazon Glacier 文件库）的 IAM 用户的影响。要在 IAM 用户的策略模拟器中使用基于资源的策略，必须在模拟中包含该资源。还必须选中该复选框以将该资源的策略包含在模拟中。

  > **Note**
  >
  > IAM 角色不支持基于资源的策略模拟。

- 如果您的 Amazon Web Services 账户 是 Amazon Organizations 中的组织的成员，则您可以测试服务控制策略（SCP）对您的基于身份的策略的影响。

  > **Note**
  >
  > 策略模拟器不会评估具有任何条件的 SCP。

- 通过将尚未附加到用户、用户组或角色的基于身份的新策略键入或复制到策略模拟器中，对这些新策略进行测试。这些仅在模拟中使用，不会保存。您不能将基于资源的策略键入或者复制到策略模拟器中。
- 使用所选服务、操作和资源测试基于身份的策略。例如，您可以通过测试确保策略允许某个实体在 Amazon S3 服务中对特定存储桶执行 `ListAllMyBuckets`、`CreateBucket` 和 `DeleteBucket` 操作。
- 通过提供上下文密钥 (如 IP 地址或日期，包含在被测试策略的 `Condition` 元素中) 模拟真实方案。

> **Note**
>
> 如果模拟中基于身份的策略没有显式检查标签的 `Condition` 元素，则策略模拟器不会模拟作为输入提供的标签。

- 识别基于身份的策略中导致允许或拒绝访问特定资源或操作的特定语句。

主题

- IAM policy simulator 的工作方式
- 使用 IAM policy simulator 所需的权限
- 使用 IAM policy simulator（控制台）
- 使用 IAM policy simulator（Amazon CLI 和 Amazon API）

IAM policy simulator 的工作方式

策略模拟器会评估基于身份的策略中的语句以及您在模拟期间提供的输入。策略模拟器的结果可能与您的实时 Amazon 环境不同。我们建议您在使用策略模拟器进行测试后，根据真实 Amazon 环境检查您的策略，以确认您获得了预期的结果。

策略模拟器在以下几方面与真实 Amazon 环境不同：

- 策略模拟器不发出真实 Amazon 服务请求，因此，您可以安全地测试在真实 Amazon 环境中会进行不需要的更改的请求。策略模拟器不考虑生产中的实际上下文键值。
- 因为策略模拟器不模拟运行所选操作，所以它不能报告对模拟请求的任何响应。返回的唯一结果是请求的操作是被允许还是被拒绝。
- 如果您在策略模拟器内编辑策略，所做的更改只会影响策略模拟器。您的 Amazon Web Services 账户 中的相应策略保持不变。

- 您不能测试具有任何条件的服务控制策略（SCP）。
- 策略模拟器不支持资源控制策略（RCP）的模拟。
- 策略模拟器不支持针对 IAM 角色和用户进行跨账户存取的模拟。

> **Note**
>
> IAM policy simulator 无法确定哪些服务支持全局条件键进行授权。例如，策略模拟器无法识别某个服务不支持 `aws:TagKeys`。

使用 IAM policy simulator 所需的权限

您可以使用策略模拟器控制台或策略模拟器 API 来测试策略。默认情况下，控制台用户可以通过将尚未附加到用户、用户组或角色的策略键入或复制到策略模拟器中来对其进行测试。这些策略仅在模拟中使用，不会揭露敏感信息。API 用户必须具有测试未附加策略的权限。您可以允许控制台或 API 用户测试附加到 Amazon Web Services 账户 中的 IAM 用户、IAM 组或角色的策略。为此，您必须提供检索这些策略的权限。要测试基于资源的策略，用户必须有权检索相关资源的策略。

有关允许用户模拟策略的控制台和 API 策略示例，请参阅the section called “示例策略：Amazon Identity and Access Management (IAM)”。

使用策略模拟器控制台所需的权限

您可以允许用户测试附加到 Amazon Web Services 账户 中的 IAM 用户、IAM 组或角色的策略。为此，您必须向用户提供检索这些策略的权限。要测试基于资源的策略，用户必须有权检索相关资源的策略。

要查看允许为附加到用户、用户组或角色的策略使用策略模拟器控制台的示例策略，请参阅 IAM：访问策略模拟器控制台。

要查看只允许具有特定路径的用户使用策略模拟器控制台的示例策略，请参阅 IAM：基于用户路径访问策略模拟器控制台。

要创建只允许为一种实体类型使用策略模拟器控制台的策略，请按照以下步骤操作。

允许控制台用户模拟用户的策略

在策略中包含以下操作：

- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetUser
- iam:GetUserPolicy
- iam:ListAttachedUserPolicies
- iam:ListGroupsForUser
- iam:ListGroupPolicies
- iam:ListUserPolicies
- iam:ListUsers

允许控制台用户模拟 IAM 组的策略

在策略中包含以下操作：

- iam:GetGroup
- iam:GetGroupPolicy
- iam:GetPolicy
- iam:GetPolicyVersion
- iam:ListAttachedGroupPolicies
- iam:ListGroupPolicies
- iam:ListGroups

允许控制台用户模拟角色的策略

在策略中包含以下操作：

- iam:GetPolicy
- iam:GetPolicyVersion
- iam:GetRole
- iam:GetRolePolicy
- iam:ListAttachedRolePolicies

- iam:ListRolePolicies
- iam:ListRoles

要测试基于资源的策略，用户必须有权检索相关资源的策略。

允许控制台用户在 Amazon S3 存储桶中测试基于资源的策略

在您的策略中包含以下操作：

- s3:GetBucketPolicy

例如，以下策略使用该操作，以允许控制台用户在特定的 Amazon S3 存储桶中模拟基于资源的策略。

JSON

```
{
        "Version":"2012-10-17",
        "Statement": [
          {
            "Effect": "Allow",
            "Action": "s3:GetBucketPolicy",
            "Resource":"arn:aws:s3:::bucket-name/*"
          }
        ]
      }
```

使用策略模拟器 API 所需的权限

策略模拟器 API 操作 GetContextKeysForCustomPolicy 和 SimulateCustomPolicy 允许用户测试尚未附加到用户、用户组或角色的策略。要测试此类策略，请将策略以字符串形式传递给 API。这些策略仅在模拟中使用，不会揭露敏感信息。您还可以使用此 API 测试附加到 Amazon Web Services 账户 中的 IAM 用户、IAM 组或角色的策略。要执行此操作，您必须为用户提供调用 GetContextKeysForPrincipalPolicy 和 SimulatePrincipalPolicy 的权限。

要查看允许为当前 Amazon Web Services 账户 中已附加策略和未附加的策略使用策略模拟器 API 的示例策略，请参阅 IAM：访问策略模拟器 API。

要创建只允许为一种策略类型使用策略模拟器 API 的策略，请按照以下步骤操作。

允许 API 用户模拟作为字符串直接传递到 API 的策略

在策略中包含以下操作：

- `iam:GetContextKeysForCustomPolicy`
- `iam:SimulateCustomPolicy`

允许 API 用户模拟附加到 IAM 用户、IAM 组、角色或资源的策略

在策略中包含以下操作：

- `iam:GetContextKeysForPrincipalPolicy`
- `iam:SimulatePrincipalPolicy`

例如，要向名为“Bob”的用户提供模拟已分配到名为“Alice”的用户的策略的权限，请向 Bob 提供访问以下资源的权限：`arn:aws:iam::777788889999:user/alice`。

要查看只允许具有特定路径的用户使用策略模拟器 API 的示例策略，请参阅 IAM：基于用户路径访问策略模拟器 API。

使用 IAM policy simulator（控制台）

默认情况下，用户可以通过将尚未附加到用户、用户组或角色的策略键入或复制到策略模拟器控制台中来对其进行测试。这些策略仅在模拟中使用，不会揭露敏感信息。

测试未附加到用户、用户组或角色的策略（控制台）

1. 在以下位置打开 IAM policy simulator 控制台：https://policysim.aws.amazon.com/。
2. 在页面顶部的 Mode: 菜单中，选择 New Policy。
3. 在 Policy Sandbox 中，选择 Create New Policy。
4. 将策略键入或复制到策略模拟器中，按照以下步骤中所述使用策略模拟器。

获得使用 IAM policy simulator 控制台的权限后，可以使用该策略模拟器测试 IAM 用户、用户组、角色或资源策略。

测试附加到用户、用户组或角色的策略（控制台）

1. 在以下位置打开 IAM policy simulator 控制台：https://policysim.aws.amazon.com/。

> **Note**
>
> 要作为 IAM 用户登录策略模拟器，请使用您的唯一登录 URL 登录 Amazon Web Services 管理控制台。然后转到https://policysim.aws.amazon.com/。有关以 IAM 用户身份登录的更多信息，请参阅 IAM 用户如何登录 Amazon。

策略模拟器将以 Existing Policies（现有策略）模式打开，并在 Users, Groups, and Roles（用户、组和角色）下列出您的账户中的 IAM 用户。

2. 选择适合于您的任务的选项：

测试目的：	执行以下操作：
附加到用户的策略	在 Users, Groups, and Roles 列表中，选择 Users。然后选择用户。
附加到用户组的策略	在 Users, Groups, and Roles 列表中，选择 Groups。然后选择用户组。
附加到角色的策略	在 Users, Groups, and Roles 列表中，选择 Roles。然后选择角色。
附加到资源的策略	请参阅Step 9。
用户、用户组或角色的自定义策略	选择 Create New Policy (创建新策略)。在新的 Policies (策略) 窗格中，键入或粘贴策略，然后选择应用。

> **提示**
>
> 要测试附加到组的策略，您可以直接从 IAM 控制台启动 IAM policy simulator：在导航窗格中，选择 User groups（用户组）。选择要对其测试策略的组的名称，然后选择权限选项卡。选择 Simulate（模拟）。
> 要测试附加到用户的客户托管策略，请执行以下操作：在导航窗格中选择 Users (用户)。选择要对其测试策略的用户的名称。然后选择 Permissions 选项卡，并展开您要测试的策略。在最右端，选择 Simulate policy。IAM policy simulator在新窗口中打开，并在 Policies（策略）窗格中显示所选策略。

3. （可选）如果您的账户是 Amazon Organizations 中某组织的成员，则请选中 Amazon Organizations SCP 旁边的复选框以将 SCP 包含在模拟评估中。SCP 是指定组织或组织单元 (OU) 的最大权限的 JSON 策略。SCP 限制成员账户中实体的权限。如果 SCP 阻止某个服务或操作，则该账户中的所有实体均无法访问该服务或执行该操作。即使管理员使用 IAM 或资源策略明确地为该服务或操作授予权限，也依然如此。

   如果您的账户不是某组织的成员，则不会显示该复选框。

4. （可选）您可以测试为 IAM 实体（用户或角色，但不包括 IAM 组）设置为权限边界的策略。如果当前为实体设置了权限边界策略，则该策略将显示在 Policies (策略) 窗格中。您仅可为实体设置一个权限边界。要测试不同的权限边界，您可以创建自定义权限边界。为此，选择 Create New Policy (创建新策略)。将打开新的 Policies (策略) 窗格。在菜单中，选择 Custom IAM Permissions Boundary Policy (自定义 IAM 权限边界策略)。输入新策略的名称，然后在下面的空间中键入或复制策略。选择应用以保存策略。接下来，选择 Back (返回) 以返回原始 Policies (策略) 窗格。然后选中要用于模拟的权限边界旁边的复选框。

5. （可选）您可以只测试附加到用户、用户组或角色的策略的子集。为此，请在策略窗格中清除每个要排除的策略旁边的复选框。

6. 在 Policy Simulator 下，选择 Select service，然后选择要测试的服务。然后选择 Select actions，选择一个或多个要测试的操作。尽管菜单一次只为一项服务显示可用选择，不过 Action Settings and Results 中会显示所选择的全部服务和操作。

7. （可选）如果在 Step 2 和 Step 5 中选择的任一策略包含具有 Amazon 全局条件键的条件，请为这些键提供值。为此，可以展开 Global Settings (全局设置) 部分，为其中显示的键名称键入值。

   > **Warning**
   >
   > 如果将某个条件键的值留空，则模拟过程会忽略该键。在某些情况下，这会导致错误，使模拟无法运行。在其他情况下，模拟运行，但结果可能不可靠。在这些情况下，模拟过程与条件键或变量包含值的真实条件不相符。

8. （可选）所选的每个操作均显示在 Action Settings and Results（操作设置和结果）列表中，在实际运行模拟前，Permission（权限）列中显示 Not simulated。在运行模拟前，您可以使用资源配置每个操作。要为特定方案配置单独的操作，请选择箭头展开相应操作的行。如果该操作支持资源级权限，您可以键入要测试其访问权限的特定资源的 Amazon Resource Name (ARN)。默认情况下，每个资源都设置为一个通配符 (*)。您还可以为任意条件上下文密钥指定值。如前所述，系统会忽略值为空的密钥，从而导致模拟失败或结果不可靠。

a. 单击操作名称旁的箭头展开每一行，可以配置准确模拟方案中的操作所需的任何其他信息。如果操作需要任何资源级权限，您可以键入要模拟访问的特定资源的 Amazon Resource Name (ARN)。默认情况下，每个资源都设置为一个通配符 (*)。

b. 如果操作支持但未设置资源级权限，可以选择 Add Resource，从而选择要添加到模拟中的资源类型。

c. 如果所选任一策略包含的 `Condition` 元素引用该操作的服务的上下文密钥，则该操作下会显示相应键名称。您可以指定在对指定的资源模拟该操作的过程中将使用的值。

需要不同资源类型组的操作

某些操作在不同情况下需要不同的资源类型。每个资源类型组都与一个方案相关联。如果其中一个适用于您的模拟，则选择它，策略模拟器需要适合该方案的资源类型。下表列出了每个受支持的方案选项，以及为运行模拟您必须定义的资源。

以下每个 Amazon EC2 场景都需要指定 `instance`、`image` 和 `security-group` 资源。如果方案包含某个 EBS 卷，您必须指定该 `volume` 作为资源。如果 Amazon EC2 场景包含 Virtual Private Cloud (VPC)，则必须提供 `network-interface` 资源。如果它包含 IP 子网，您必须指定 `subnet` 资源。有关 Amazon EC2 场景选项的更多信息，请参阅 Amazon EC2 用户指南中的支持的平台。

- EC2-VPC-InstanceStore

  实例、映像、安全组、网络接口

- EC2-VPC-InstanceStore-Subnet

  实例、映像、安全组、网络接口、子网

- EC2-VPC-EBS

  实例、映像、安全组、网络接口、卷

- EC2-VPC-EBS-Subnet

  实例、映像、安全组、网络接口、子网、卷

9. (可选) 如果要在模拟中包含基于资源的策略，必须首先在 Step 6 中选择要对资源模拟的操作。展开所选操作的行，对要模拟的策略键入该资源的 ARN。然后选择 ARN 文本框旁边的 Include Resource Policy（包括资源策略）。IAM policy simulator 当前仅支持来自以下服务的基于资源的策略：Amazon S3（仅限基于资源的策略；当前不支持 ACL）、Amazon SQS、Amazon SNS 和未锁定的 Amazon Glacier 文件库（当前不支持锁定的文件库）。

10. 选择右上角的 Run Simulation。

    Action Settings and Results 每行中的 Permission 列显示对指定资源模拟该操作的结果。

11. 要查看策略中的哪个语句显式允许或拒绝操作，请在 Permission（权限）列中，选择 ***N*** matching statement(s) 链接，展开该行。然后，选择 Show statement (显示语句) 链接。Policies 窗格显示其语句会影响突出显示的模拟结果的相关策略。

> **Note**
>
> 如果操作为隐式拒绝，意即，如果操作仅因未明确允许而被拒绝，则 List（列表）和 Show statement（显示语句）选项不会显示。

IAM policy simulator 控制台消息故障排除

下表列出了在使用 IAM policy simulator 时可能遇到的告知性和警告性消息。表中还提供了解决消息所示问题的操作步骤。

Message	解决步骤
This policy has been edited。Changes will not be saved to your account。	无需操作。 此为告知性消息。如果在 IAM policy simulator 中编辑现有策略，则所做的更改不会影响您的 Amazon Web Services 账户。通过策略模拟器，您可以仅出于测试目的对策略进行更改。
无法获取资源策略。原因：######	策略模拟器无法访问所请求的基于资源的策略。确保指定的资源 ARN 正确，并且运行模拟的用户有权读取该资源的策略。
One or more policies require values in the simulation settings。The simulation might fail without these values。	如果您测试的策略中包含条件键或变量，但未在 Simulation Settings (模拟设置) 中为这些键或变量提供任何值，则会显示此消息。 要取消显示此消息，请选择 Simulation Settings (模拟设置)，然后为每个条件键或变量输入值。

Message	解决步骤
You have changed policies。These results are no longer valid。	若您在 Results 窗格中显示结果的同时更改了所选策略，则会显示此消息。Results 窗格中显示的结果不会动态更新。 要取消显示此消息，请再次选择 Run Simulation，以根据 Policies 窗格中所做的更改显示新的模拟结果。
The resource you typed for this simulation does not match this service。	如果您在 Simulation Settings（模拟设置）窗格中键入的 Amazon Resource Name (ARN) 与您为当前模拟选择的服务不相符，则会显示此消息。例如，如果为 Amazon DynamoDB 资源指定一个 ARN，但选择 Amazon Redshift 作为要模拟的服务，则会显示此消息。 要取消显示此消息，请执行下列操作之一： • 请从 Simulation Settings 窗格中的框中删除该 ARN。 • 选择与在 Simulation Settings 中指定的 ARN 相符的服务。
此操作所属的服务除支持基于资源的策略之外，还支持特殊访问控制机制，如 Amazon S3 ACL 或 Amazon Glacier 文件库锁定策略。策略模拟器不支持这些机制，因此结果可能与您的生产环境不同。	无需操作。 此为告知性消息。在当前版本中，策略模拟器评估附加到用户和 IAM 组的策略，可评估针对 Amazon S3、Amazon SQS、Amazon SNS 和 Amazon Glacier 的基于资源的策略。策略模拟器只支持其他 Amazon 服务支持的部分访问控制机制。

Message	解决步骤
DynamoDB FGAC is currently not supported。	无需操作。 该信息性消息指的是精细访问控制。精细的访问控制是指使用 IAM policy 条件来确定谁可以访问 DynamoDB 表和索引中的各个数据项和属性的功能。它还指可以对这些表和索引执行的操作。IAM policy simulator 的当前版本不支持这类策略条件。有关 DynamoDB 访问权限精细控制的更多信息，请参阅 DynamoDB 访问权限的精细控制。
You have policies that do not comply with the policy syntax。您可以使用策略模拟器查看您策略的推荐更新。	如果策略不符合 IAM policy 语法，则会在策略列表的顶部显示此消息。要模拟这些策略，请查看位于 IAM 策略验证 的策略验证选项来识别和修复这些策略。
This policy must be updated to comply with the latest policy syntax rules。	如果您的策略不符合 IAM policy 语法，则会显示此消息。要模拟这些策略，请查看位于 IAM 策略验证 的策略验证选项来识别和修复这些策略。

使用 IAM policy simulator（Amazon CLI 和 Amazon API）

策略模拟器命令通常需要调用 API 操作以执行两项操作：

1. 评估策略并返回策略引用的上下文密钥的列表。您需要了解引用了哪些上下文密钥，以便在下一个步骤中提供它们的值。
2. 模拟策略，在模拟过程中提供操作、资源和上下文密钥的列表。

出于安全原因，API 操作已分为两个组：

- 只对以字符串形式直接传递给 API 的策略进行模拟的 API 操作。此组包括 GetContextKeysForCustomPolicy 和 SimulateCustomPolicy。
- 对挂载到指定 IAM 用户、用户组、角色或资源的策略进行模拟的 API 操作。因为这些 API 操作可以获得分配给其他 IAM 实体的权限的详细信息，所以您应考虑限制对这些 API 操作的访问。此组包括

GetContextKeysForPrincipalPolicy 和 SimulatePrincipalPolicy。有关限制对 API 操作的访问的更多信息，请参阅示例策略：Amazon Identity and Access Management (IAM)。

在这两种情况下，API 操作都将模拟一个或多个策略对操作和资源的列表的影响。每个操作都与每个资源配对，模拟将确定策略对该资源是允许还是拒绝该操作。您还可以为您的策略引用的任何上下文密钥提供值。通过首先调用 GetContextKeysForCustomPolicy 或 GetContextKeysForPrincipalPolicy，可以获取策略引用的上下文密钥的列表。如果不提供上下文密钥值，模拟仍会运行。但结果可能不可靠，这是因为策略模拟器无法在评估中包含该上下文密钥。

获取上下文密钥的列表（Amazon CLI、Amazon API）

使用以下命令评估策略列表，返回策略中使用的上下文密钥的列表。

- Amazon CLI`aws iam get-context-keys-for-custom-policy` 和 `aws iam get-context-keys-for-principal-policy`
- Amazon API：GetContextKeysForCustomPolicy 和 GetContextKeysForPrincipalPolicy

模拟 IAM policy（Amazon CLI、Amazon API）

使用以下命令模拟 IAM policy，以确定用户的有效权限。

- Amazon CLI`aws iam simulate-custom-policy` 和 `aws iam simulate-principal-policy`
- Amazon API：SimulateCustomPolicy 和 SimulatePrincipalPolicy

添加和删除 IAM 身份权限

可使用策略定义身份（用户、用户组或角色）的权限。您可以通过使用 Amazon Web Services 管理控制台、Amazon Command Line Interface (Amazon CLI) 或 IAM API 附加和分离 Amazon 策略来添加和删除权限。您还可使用策略为使用相同方法的唯一实体（用户或角色）设置 permissions boundaries（权限边界）。权限边界是一项控制实体可拥有的最大权限的高级 Amazon 功能。

主题

- 术语
- 查看身份活动

- 添加 IAM 身份权限（控制台）
- 删除 IAM 身份权限（控制台）
- 添加 IAM policy (Amazon CLI)
- 移除 IAM policy (Amazon CLI)
- 添加 IAM policy (Amazon API)
- 删除 IAM policy (Amazon API)

术语

将权限策略与身份（IAM 用户、IAM 组和 IAM 角色）相关联时，术语和过程会因使用的是托管式策略还是内联策略而异：

- 附加 - 与托管策略一起使用。您将托管策略附加到身份（用户、用户组或角色）。如果附加一个策略，则会将该策略中的权限应用于身份。
- 分离 - 与托管策略一起使用。将托管策略与 IAM 身份（用户、用户组或角色）分离。如果分离一个策略，则会将其权限从身份中删除。
- 嵌入 - 与内联策略一起使用。您在身份（用户、用户组或角色）中嵌入内联策略。如果嵌入一个策略，则会将该策略中的权限应用于身份。由于内联策略存储在身份中，因此，将嵌入而不是附加该策略，尽管结果类似。

> **Note**
>
> 您只能将服务相关角色的内联策略嵌入依赖该角色的服务。请参阅服务的 Amazon 文档，以了解它是否支持该功能。

- 删除 - 与内联策略一起使用。从 IAM 身份（用户、用户组或角色）删除内联策略。如果删除一个策略，则会将其权限从身份中删除。

> **Note**
>
> 您只能将服务相关角色的内联策略从依赖该角色的服务中删除。请参阅服务的 Amazon 文档，以了解它是否支持该功能。

您可以使用控制台、Amazon CLI 或 Amazon API 来执行这些操作中的任一操作。

更多信息

- 有关托管策略和内联策略之间的差别的更多信息，请参阅托管策略与内联策略。
- 有关权限边界的更多信息，请参阅IAM 实体的权限边界。
- 有关 IAM policy 的一般信息，请参阅。Amazon Identity and Access Management 中的策略和权限
- 有关验证 IAM 用户策略的信息，请参阅 IAM 策略验证。
- Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

查看身份活动

在更改身份（用户、用户组或角色）的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

添加 IAM 身份权限（控制台）

您可以使用 Amazon Web Services 管理控制台向身份（用户、用户组或角色）添加权限。为此，请附加控制权限的托管策略，或指定充当权限边界的策略。您还可以嵌入内联策略。

将托管策略用作身份的权限策略（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选中要附加的策略的名称旁边的单选按钮。您可以使用搜索框筛选策略列表。
4. 选择 Actions（操作），然后选择 Attach（附加）。
5. 选择一个或多个要将策略附加到的身份。您可以使用搜索框来筛选主体实体列表。在选择身份后，选择附加策略。

使用托管策略设置权限边界（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。

3. 在策略列表中，选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略详细信息页面上，选择附加的实体选项卡，然后（如有必要）打开附加为权限边界部分并选择将策略设置为权限边界。
5. 选择要将策略用于其权限边界的一个或多个用户或角色。您可以使用搜索框来筛选主体实体列表。选择主体后，选择设置权限边界。

为用户或角色嵌入内联策略（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择用户或角色。
3. 在列表中，选择要在其中嵌入策略的用户或角色的名称。
4. 选择权限选项卡。
5. 选择添加权限，然后选择创建内联策略。

> **Note**
>
> 您不能在 IAM 中的服务相关角色中嵌入内联策略。由于链接服务定义了您是否能修改角色的权限，因此，您可能能够从服务控制台、API 或 Amazon CLI 添加其他策略。要查看某个服务的服务相关角色文档，请参阅使用 IAM 的 Amazon 服务并在您的服务的 Service-Linked Role 列中选择 Yes。

6. 选择以下方法之一来查看创建策略所需的步骤：
   - 导入现有托管策略 - 您可以在您的账户中导入一个托管策略，然后编辑该策略以根据您的特定要求进行自定义。托管策略可以是 Amazon 托管策略，也可以是您以前创建的客户托管策略。
   - 使用可视化编辑器创建策略 - 您可以在可视化编辑器中从头开始构建新的策略。如果使用可视化编辑器，您不需要了解 JSON 语法。
   - 使用 JSON 编辑器创建策略 – 在 JSON 编辑器选项中，您可以使用 JSON 语法创建策略。您可以键入新的 JSON 策略文档或粘贴示例策略。
7. 创建内联策略后，它会自动嵌入您的用户和角色。

为用户组嵌入内联策略（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 User groups（用户组）。
3. 在列表中，请选择要在其中嵌入策略的用户组的名称。
4. 请选择 Permissions（权限）选项卡，然后选择 Add permissions（添加权限），接着选择 Create inline policy（创建内联策略）。
5. 请执行以下操作之一：
   - 选择可视化选项以创建策略。有关更多信息，请参阅 使用可视化编辑器创建策略。
   - 选择 JSON 选项以创建策略。有关更多信息，请参阅 使用 JSON 编辑器创建策略。
6. 若您满意所创建的策略，请选择创建策略。

更改一个或多个实体的权限边界（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略详细信息页面上，选择附加的实体选项卡，然后（如有必要）打开附加为权限边界部分。选中要更改其边界的用户或角色旁边的复选框，然后选择更改。
5. 选择要用于权限边界的新策略。您可以使用搜索框筛选策略列表。选择策略后，选择设置权限边界。

删除 IAM 身份权限（控制台）

您可以使用 Amazon Web Services 管理控制台删除身份（用户、用户组或角色）的权限。为此，请分离控制权限的托管策略，或删除充当权限边界的策略。您还可以删除内联策略。

分离用作权限策略的托管策略（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。

3. 在策略列表中，选中要分离的策略的名称旁边的单选按钮。您可以使用搜索框筛选策略列表。
4. 选择 Actions（操作），然后选择 Detach（删除）。
5. 选择要从中分离策略的身份。您可以使用搜索框筛选身份列表。在选择身份后，选择分离策略。

删除权限边界（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要设置的策略的名称。您可以使用搜索框筛选策略列表。
4. 在策略摘要页面上，选择附加的实体选项卡，然后（如有必要）打开附加为权限边界部分并选择要移除权限边界的实体。然后选择移除边界。
5. 确认您要移除边界并选择移除边界。

删除内联策略（控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups（组）、Users（用户）或 Roles（角色）。
3. 在列表中，请选择具有要修改的策略的用户组、用户或角色的名称。
4. 选择权限选项卡。
5. 选中策略旁边的复选框，然后选择移除。
6. 在确认对话框中，选择移除。

添加 IAM policy (Amazon CLI)

您可以使用 Amazon CLI向身份（用户、用户组或角色）添加权限。为此，请附加控制权限的托管策略，或指定充当权限边界的策略。您还可以嵌入内联策略。

将托管策略用作实体的权限策略 (Amazon CLI)

1. （可选）要查看有关托管策略的信息，请运行以下命令：
   - 列出托管策略：aws iam list-policies

- 检索有关托管策略的详细信息：get-policy

2. 要将托管策略附加到身份（用户、用户组或角色），请使用以下命令之一：

- aws iam attach-user-policy
- aws iam attach-group-policy
- aws iam attach-role-policy

使用托管策略设置权限边界 (Amazon CLI)

1. （可选）要查看有关托管策略的信息，请运行以下命令：

- 列出托管策略：aws iam list-policies
- 检索有关托管策略的详细信息：aws iam get-policy

2. 要使用托管策略为实体（用户或角色）设置权限边界，请使用下列命令之一：

- aws iam put-user-permissions-boundary
- aws iam put-role-permissions-boundary

嵌入内联策略 (Amazon CLI)

要将内联策略嵌入身份（用户、用户组或不是服务相关角色的角色），请使用以下命令之一：

- aws iam put-user-policy
- aws iam put-group-policy
- aws iam put-role-policy

移除 IAM policy (Amazon CLI)

您可以使用 Amazon CLI 分离控制权限的托管策略，或删除充当权限边界的策略。您还可以删除内联策略。

分离用作权限策略的托管策略 (Amazon CLI)

1. （可选）要查看有关策略的信息，请运行以下命令：

- 列出托管策略：aws iam list-policies
- 检索有关托管策略的详细信息：aws iam get-policy

2. （可选）要了解策略与身份之间的关系，请运行以下命令：
   - 列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色）：
     - aws iam list-entities-for-policy
   - 要列出附加到身份（用户、用户组或角色）的托管策略，请使用以下一个命令：
     - aws iam list-attached-user-policies
     - aws iam list-attached-group-policies
     - aws iam list-attached-role-policies
3. 要从身份（用户、用户组或角色）分离托管策略，请使用以下命令之一：
   - aws iam detach-user-policy
   - aws iam detach-group-policy
   - aws iam detach-role-policy

删除权限边界 (Amazon CLI)

1. （可选）要查看当前用于为用户或角色设置权限边界的托管策略，请运行以下命令：
   - aws iam get-user
   - aws iam get-role
2. （可选）要查看将托管策略用于其权限边界的用户或角色，请运行以下命令：
   - aws iam list-entities-for-policy
3. （可选）要查看有关托管策略的信息，请运行以下命令：
   - 列出托管策略：aws iam list-policies
   - 检索有关托管策略的详细信息：aws iam get-policy
4. 要从用户或角色删除权限边界，请使用以下命令之一：
   - aws iam delete-user-permissions-boundary
   - aws iam delete-role-permissions-boundary

删除内联策略 (Amazon CLI)

1. （可选）要列出附加到身份（用户、用户组和角色）的所有内联策略，请使用以下命令之一：

- aws iam list-user-policies
- aws iam list-group-policies
- aws iam list-role-policies

2. （可选）要检索嵌入到身份（用户、用户组或角色）中的内联策略文档，请使用以下命令之一：

- aws iam get-user-policy
- aws iam get-group-policy
- aws iam get-role-policy

3. 要从身份（用户、用户组或不是服务相关角色的角色）中删除内联策略，请使用以下命令之一：

- aws iam delete-user-policy
- aws iam delete-group-policy
- aws iam delete-role-policy

添加 IAM policy (Amazon API)

您可以使用 Amazon API 附加控制权限的托管策略，或指定充当权限边界的策略。您还可以嵌入内联策略。

将托管策略用作实体的权限策略 (Amazon API)

1. （可选）要查看有关策略的信息，请调用以下操作：

- 列出托管策略：ListPolicies
- 检索有关托管策略的详细信息：GetPolicy

2. 要将托管策略附加到身份（用户、用户组或角色），请调用以下操作之一：

- AttachUserPolicy
- AttachGroupPolicy
- AttachRolePolicy

使用托管策略设置权限边界 (Amazon API)

1. （可选）要查看有关托管策略的信息，请调用以下操作：

- 列出托管策略：ListPolicies

- 检索有关托管策略的详细信息：GetPolicy

2. 要使用托管策略为实体（用户或角色）设置权限边界，请调用下列操作之一：

- PutUserPermissionsBoundary
- PutRolePermissionsBoundary

嵌入内联策略 (Amazon API)

要将内联策略嵌入身份（用户、用户组或不是服务相关角色的角色），请调用以下操作之一：

- PutUserPolicy
- PutGroupPolicy
- PutRolePolicy

删除 IAM policy (Amazon API)

您可以使用 Amazon API 分离控制权限的托管策略，或删除充当权限边界的策略。您还可以删除内联策略。

分离用作权限策略的托管策略 (Amazon API)

1. （可选）要查看有关策略的信息，请调用以下操作：

- 列出托管策略：ListPolicies
- 检索有关托管策略的详细信息：GetPolicy

2. （可选）要了解策略与身份之间的关系，请调用以下操作：

- 列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色）：
  - ListEntitiesForPolicy
- 要列出附加到身份（用户、用户组或角色）的托管策略，请调用以下一项操作：
  - ListAttachedUserPolicies
  - ListAttachedGroupPolicies
  - ListAttachedRolePolicies

3. 要将托管策略与身份（用户、用户组或角色）分离，请调用以下操作之一：

- DetachUserPolicy

- DetachGroupPolicy
- DetachRolePolicy

删除权限边界 (Amazon API)

1. （可选）要查看当前用于为用户或角色设置权限边界的托管策略，请调用以下操作：
   - GetUser
   - GetRole
2. （可选）要查看将托管策略用于其权限边界的用户或角色，请调用以下操作：
   - ListEntitiesForPolicy
3. （可选）要查看有关托管策略的信息，请调用以下操作：
   - 列出托管策略：ListPolicies
   - 检索有关托管策略的详细信息：GetPolicy
4. 要从用户或角色删除权限边界，请调用以下操作之一：
   - DeleteUserPermissionsBoundary
   - DeleteRolePermissionsBoundary

删除内联策略 (Amazon API)

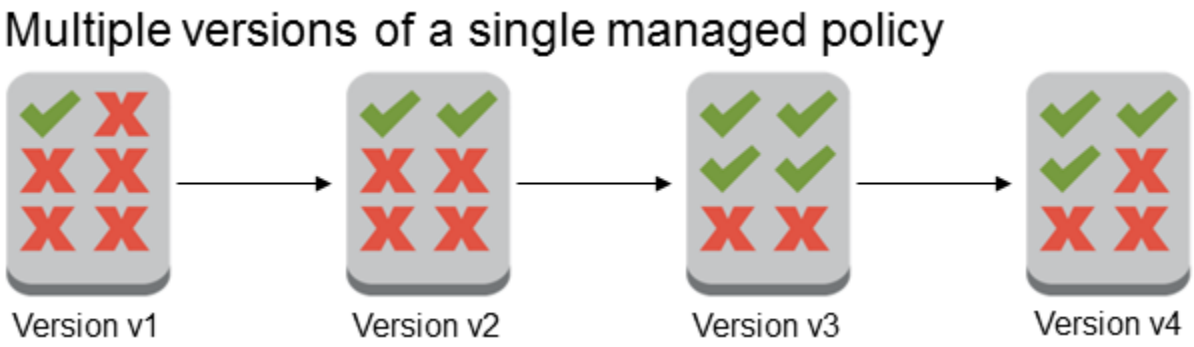
1. （可选）要列出附加到身份（用户、用户组和角色）的所有内联策略，请调用以下操作之一：
   - ListUserPolicies
   - ListGroupPolicies
   - ListRolePolicies
2. （可选）要检索嵌入到身份（用户、用户组或角色）中的内联策略文档，请调用以下操作之一：
   - GetUserPolicy
   - GetGroupPolicy
   - GetRolePolicy
3. 要将内联策略从身份（用户、用户组或不是服务相关角色的角色）中删除，请调用以下操作之一：
   - DeleteUserPolicy

- DeleteGroupPolicy
- DeleteRolePolicy

IAM 策略版本控制

在对 IAM 客户管理型策略进行更改时，以及 Amazon 对 Amazon 托管策略进行更改时，更改的策略不会覆盖现有的策略。而是由 IAM 创建新的托管策略版本。IAM 最多可以存储五个版本的客户管理策略。IAM 不支持内联策略版本控制。

下图说明了客户管理型策略的版本控制。在此示例中，已保存版本 1-4。您最多可以将五个托管策略版本保存到 IAM。在您编辑将创建第六个已保存版本的策略时，可以选择不再保存哪个较旧版本。您可以随时恢复到其他四个已保存版本中的任何一个。



策略版本与 `Version` 策略元素不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。要了解 `Version` 策略元素的更多信息，请参阅IAM JSON 策略元素：Version。

您可以使用不同版本跟踪对托管策略的更改。例如，您对托管策略进行更改，然后发现更改有意外效果。在这种情况下，通过将以前版本设置为默认 版本，可以回滚到以前版本的托管策略。

下面的主题说明了如何对托管式策略进行版本控制。

主题

- 版本限制
- 使用不同版本回滚更改
- 用于设置默认策略版本的权限
- 设置客户托管策略的默认版本

版本限制

一个托管策略最多可以有五个版本。如果需要从 Amazon Command Line Interface 或 Amazon API 在五个版本之外更改托管策略更改，必须先删除一个或多个现有版本。如果使用的是 Amazon Web Services 管理控制台，则您在编辑策略之前不必删除版本。当您保存第六个版本时，会出现一个对话框，提示您删除策略的一个或多个非默认版本。您可以查看每个版本的 JSON 策略文档，以方便做出决定。有关该对话框的详细信息，请参阅the section called “编辑 IAM 策略”。

您可以根据需要删除任何托管策略版本 (默认版本除外)。删除某个版本时，其余版本的版本标识符不会更改。因此，版本标识符可能是不连续的。例如，如果您删除托管策略版本 v2 和 v4，然后添加两个新版本，则其余版本标识符可能是 v1、v3、v5、v6 和 v7。

使用不同版本回滚更改

您可以设置客户管理型策略的默认版本以回滚您的更改。例如，考虑以下情形：

您创建一个客户管理型策略，以允许用户使用 Amazon Web Services 管理控制台 管理特定的 Amazon S3 存储桶。创建时，您的客户管理型策略只有一个版本（标识为 v1），因此该版本自动设置为默认版本。该策略可正常工作。

随后，您将更新策略，以添加权限对第二个 Amazon S3 存储桶进行管理。IAM 会创建新的策略版本（标识为 v2），该版本将包含您所做的更改。您将版本 v2 设置为默认版本，不久之后，您的用户报告，他们缺少使用 Amazon S3 控制台的权限。在此情况下，您可以回滚到策略版本 v1，您知道该版本可正常工作。为此，您可以将版本 v1 设置为默认版本。您的用户现在可以使用 Amazon S3 控制台来管理原始存储桶。

后来，在确定策略版本 v2 中的错误之后，您再次更新策略，添加了对第二个 Amazon S3 存储桶的管理权限。IAM 会创建另一个新版本的策略，标识为 v3。您将版本 v3 设置为默认版本，此版本可正常工作。此时，您删除策略版本 v2。

用于设置默认策略版本的权限

设置默认策略版本所需的权限对应于任务的 Amazon API 操作。您可以使用 `CreatePolicyVersion` 或 `SetDefaultPolicyVersion` API 操作来设置默认策略版本。要允许某人设置现有策略的默认策略版本，您可以允许访问 `iam:CreatePolicyVersion` 操作或 `iam:SetDefaultPolicyVersion` 操作。`iam:CreatePolicyVersion` 操作允许他们创建新版本的策略并将该版本设置为默认版本。`iam:SetDefaultPolicyVersion` 操作允许他们将任何现有策略版本设置为默认版本。

> **Important**
>
> 要防止用户更改策略的默认版本，必须同时拒绝 `iam:CreatePolicyVersion` 和 `iam:SetDefaultPolicyVersion`。

您可以使用以下策略来拒绝用户访问以更改现有客户管理型策略：

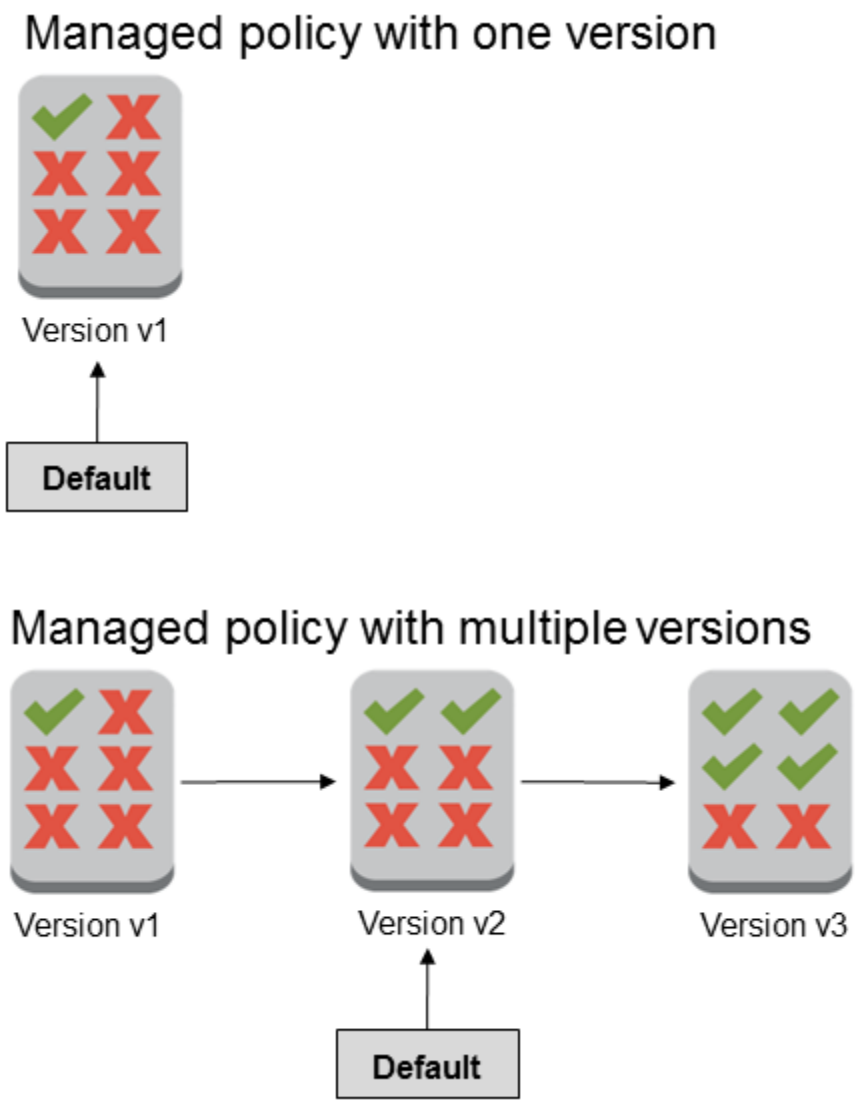
JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action": [
                "iam:CreatePolicyVersion",
                "iam:SetDefaultPolicyVersion"
            ],
            "Resource": "arn:aws:iam::*:policy/POLICY-NAME"
        }
    ]
}
```

设置客户托管策略的默认版本

托管策略的一个版本会设置为默认 版本。策略的默认版本是有效版本，即对托管式策略附加到的所有主体实体（IAM 用户、IAM 组和 IAM 角色）都生效的版本。

创建客户管理型策略时，策略从标识为 v1 的单个版本开始。对于只有单个版本的托管策略，该版本会自动设置为默认版本。对于有多个版本的客户托管策略，由您选择要设置为默认版本的版本。对于 Amazon 托管策略，默认版本由 Amazon 设置。下面的示意图对此概念进行说明。



您可以设置客户管理型策略的默认版本，以将该版本应用于该策略附加到的每个 IAM 身份（用户、用户组和角色）。您无法为 Amazon 托管策略或内联策略设置默认版本。

设置客户管理型策略的默认版本（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要设置其默认版本的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要设置为默认版本的版本旁边的复选框，然后选择设为默认。

要了解如何从 Amazon Command Line Interface 或 Amazon API 中设置客户管理型策略的默认版本，请参阅编辑 IAM 策略（Amazon CLI）。

编辑 IAM 策略

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。策略作为 JSON 文档存储在 Amazon 中，并在 IAM 中作为基于身份的策略附加到主体。您可以将基于身份的策略附加到主体（或身份），例如，IAM 用户组、用户或角色。基于身份的策略包括 Amazon 托管策略、客户托管策略和内联策略。您可以在 IAM 中编辑客户托管策略和内联策略。Amazon 托管策略无法编辑。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

通常来说，最好使用客户管理型策略而非内联策略或 Amazon 托管策略。Amazon 托管策略通常提供广泛的管理权限或只读权限。内联策略不能在其他身份上重复使用，也不能在其所存在的身份之外托管。为了获得最高安全性，请授予最低权限，这意味着仅授予执行特定作业任务所需的权限。

创建或编辑 IAM policy 时，Amazon 可以自动执行策略验证，以帮助您创建具有最低权限的有效策略。在 Amazon Web Services 管理控制台 中，IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

您可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 在 IAM 中编辑客户管理型策略和内联策略。有关使用 Amazon CloudFormation 模板添加或更新策略的更多信息，请参阅《Amazon CloudFormation 用户指南》中的 Amazon Identity and Access Management 资源类型参考。

主题

- 编辑 IAM 策略（控制台）
- 编辑 IAM 策略（Amazon CLI）
- 编辑 IAM 策略（Amazon API）

编辑 IAM 策略（控制台）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon Web Services 管理控制台 在 IAM 中编辑客户管理型策略和内联策略。Amazon 托管策略无法编辑。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

有关策略结构和语法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限和IAM JSON 策略元素参考。

先决条件

在更改策略的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

编辑客户托管策略（控制台）

您可以从 Amazon Web Services 管理控制台 中编辑客户管理型策略以更改在策略中定义的权限。客户托管策略最多可以具有五个版本。这是非常重要的，因为如果在五个版本以外更改托管策略，Amazon Web Services 管理控制台将提示您决定删除哪个版本。也可以更改策略的默认版本，或者在编辑之前删除一个版本以避免出现提示。要了解版本的更多信息，请参阅IAM 策略版本控制。

Console

编辑客户管理型策略

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要编辑的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择权限选项卡，然后选择编辑。
5. 请执行以下操作之一：

   - 选择可视化选项以更改您的策略，而无需了解 JSON 语法。您可以更改策略中的每个权限块的服务、操作、资源或可选条件。也可以导入一个策略以在您的策略底部添加其他权限。完成更改后，选择下一步以继续。
   - 选择 JSON 选项，然后在 JSON 文本框中键入或粘贴文本以修改您的策略。也可以导入一个策略以在您的策略底部添加其他权限。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

6. 在查看并保存页面上，查看此策略中定义的权限，然后选择保存更改以保存您的工作。
7. 如果管理型策略已达到最大版本数（5 个），选择保存更改将显示对话框。要保存您的新版本，策略最旧的非默认版本将被移除并替换为该新版本。（可选）您也可以将新版本设置为策略的默认版本。

   选择保存更改以保存您的新策略版本。

设置客户管理型策略的默认版本（控制台）

您可以从 Amazon Web Services 管理控制台 中设置客户管理型策略的默认版本。您可以使用该策略为整个组织中的权限建立一致的基准配置。该策略的所有新附件都将使用这组标准化的权限。

Console

设置客户托管策略的默认版本（控制台）

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要设置其默认版本的策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要设置为默认版本的版本旁的复选框，然后选择 Set as default。

删除某个版本的客户管理型策略（控制台）

您可能需要删除客户管理型策略的某个版本，才能删除不再需要的权限或存在潜在安全风险的过时权限或不正确权限。仅维护必要的版本有助于确保始终处于五个管理型策略版本的限制范围内，从而为将来的更新和改进留出空间。您可以从 Amazon Web Services 管理控制台 中删除客户管理型策略的某个版本。

Console

删除客户管理型策略的某个版本

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 选择要删除其版本的客户托管策略的名称。您可以使用搜索框筛选策略列表。
4. 选择 Policy Versions (策略版本) 选项卡。选中要删除的版本旁边的复选框，然后选择删除。
5. 确认您要删除该版本，然后选择 Delete。

编辑内联策略（控制台）

您可能需要编辑客户管理型策略以更新或完善所授予的权限，确保这些权限与组织不断变化的安全要求和访问控制需求保持一致。您可通过编辑来调整策略的 JSON 文档，并添加、修改或删除特定操作、资源或条件，以维护最低权限原则并适应环境或流程的变化。您可以从 Amazon Web Services 管理控制台编辑内联策略。

Console

编辑用户、用户组或角色的内联策略

1. 在导航窗格中，选择 Users（用户）、User groups（用户组）或 Roles（角色）。
2. 请选择具有要修改的策略的用户组、用户或角色的名称。然后选择权限选项卡并展开此策略。
3. 要编辑内联策略，请选择 Edit Policy。
4. 请执行以下操作之一：
   - 选择可视化选项以更改您的策略，而无需了解 JSON 语法。您可以更改策略中的每个权限块的服务、操作、资源或可选条件。也可以导入一个策略以在您的策略底部添加其他权限。完成更改后，选择下一步以继续。
   - 选择 JSON 选项，然后在 JSON 文本框中键入或粘贴文本以修改您的策略。也可以导入一个策略以在您的策略底部添加其他权限。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。要在不影响当前附加的实体的情况下保存更改，请清除 Save as default version 的复选框。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

5. 在审核页面上，审核策略摘要，然后选择保存更改进行保存。

编辑 IAM 策略（Amazon CLI）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon Command Line Interface（Amazon CLI）在 IAM 中编辑客户管理型策略和内联策略。Amazon 托管策略无法编辑。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

有关策略结构和语法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限和IAM JSON 策略元素参考。

先决条件

在更改策略的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

编辑客户托管策略 (Amazon CLI)

您可以从 Amazon CLI 编辑客户托管策略。

> **Note**
>
> 一个托管策略最多可以有五个版本。如果您需要在五个版本之外继续对客户托管策略进行更改，则必须首先删除一个或多个现有版本。

编辑客户托管策略 (Amazon CLI)

1. （可选）要查看有关策略的信息，请运行以下命令：

   - 列出托管策略：list-policies

- 检索有关托管策略的详细信息：get-policy

2. （可选）要了解策略与身份之间的关系，请运行以下命令：

   - 列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色）：
     - list-entities-for-policy
   - 列出附加到身份（用户、用户组或角色）的托管策略：
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies

3. 要编辑客户托管策略，请运行以下命令：

   - create-policy-version

4. （可选）要验证客户托管策略，请运行以下 IAM Access Analyzer 命令：

   - validate-policy

设置客户管理型策略的默认版本（Amazon CLI）

您可以从 Amazon CLI 中设置客户管理型策略的默认版本。

设置客户托管策略的默认版本 (Amazon CLI)

1. （可选）要列出托管策略，请运行以下命令：

   - list-policies

2. 要设置客户托管策略的默认版本，请运行以下命令：

   - set-default-policy-version

删除客户管理型策略的某个版本（Amazon CLI）

您可以从 Amazon CLI 中删除客户管理型策略的某个版本。

删除客户托管策略的某个版本 (Amazon CLI)

1. （可选）要列出托管策略，请运行以下命令：

   - list-policies

2. 要删除客户托管策略，请运行以下命令：

   - delete-policy-version

编辑内联策略（Amazon CLI）

您可以从 Amazon CLI编辑内联策略。

要编辑内联策略（Amazon CLI）

1. （可选）要查看有关策略的信息，请运行以下命令：

   - 要列出与身份（用户、用户组或角色）关联的内联策略：
     - list-user-policies
     - list-role-policies
     - list-group-policies
   - 要检索有关内联策略的详细信息：
     - get-user-policy
     - get-role-policy
     - get-group-policy
2. 要编辑内联策略，请运行以下命令：

   - put-user-policy
   - put-role-policy
   - put-group-policy
3. （可选）要验证内联策略，请运行以下 IAM Access Analyzer 命令：

   - validate-policy

编辑 IAM 策略（Amazon API）

策略是一个实体；在附加到身份或资源时，策略定义了它们的权限。您可以使用 Amazon API 在 IAM 中编辑客户管理型策略和内联策略。Amazon 托管策略无法编辑。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

有关策略结构和语法的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限和IAM JSON 策略元素参考。

先决条件

在更改策略的权限之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

编辑客户托管策略 (Amazon API)

您可以使用 Amazon API 编辑客户托管策略。

> **Note**
>
> 一个托管策略最多可以有五个版本。如果您需要在五个版本之外继续对客户托管策略进行更改，则必须首先删除一个或多个现有版本。

编辑客户托管策略 (Amazon API)

1. （可选）要查看有关策略的信息，请调用以下操作：
   - 列出托管策略：ListPolicies
   - 检索有关托管策略的详细信息：GetPolicy
2. （可选）要了解策略与身份之间的关系，请调用以下操作：
   - 列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色）：
     - ListEntitiesForPolicy
   - 列出附加到身份（用户、用户组或角色）的托管策略：
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies
3. 要编辑客户托管策略，请调用以下操作：
   - CreatePolicyVersion
4. （可选）要验证客户托管策略，请调用以下 IAM Access Analyzer 操作：
   - ValidatePolicy

设置客户管理型策略的默认版本（Amazon API）

您可以从 Amazon API 中设置客户管理型策略的默认版本。

设置客户托管策略的默认版本 (Amazon API)

1. （可选）要列出托管策略，请调用以下操作：
   - ListPolicies
2. 要设置客户托管策略的默认版本，请调用以下操作：
   - SetDefaultPolicyVersion

删除客户管理型策略的某个版本（Amazon API）

您可以从 Amazon API 中删除客户管理型策略的某个版本。

删除客户托管策略的某个版本 (Amazon API)

1. （可选）要列出托管策略，请调用以下操作：
   - ListPolicies
2. 要删除客户托管策略，请调用以下操作：
   - DeletePolicyVersion

编辑内联策略（Amazon API）

您可以从 Amazon API 中编辑内联策略。

要编辑内联策略（Amazon API）

1. （可选）要查看有关内联策略的信息，请调用以下操作：
   - 要列出与身份（用户、用户组或角色）关联的内联策略：
     - ListUserPolicies
     - ListRolePolicies
     - ListGroupPolicies
   - 要检索有关内联策略的详细信息：

- GetUserPolicy
- GetRolePolicy
- GetGroupPolicy

2. 要编辑内联策略，请调用以下操作：

- PutUserPolicy
- PutRolePolicy
- PutGroupPolicy

3. （可选）要验证内联策略，请运行以下 IAM Access Analyzer 操作：

- ValidatePolicy

删除 IAM 策略

您可以使用 Amazon Web Services 管理控制台、Amazon Command Line Interface（Amazon CLI）或 Amazon API 删除 IAM 策略。

> **Note**
>
> 对 IAM policy 的删除是永久性的。策略删除后，无法恢复。

有关 IAM 策略结构和语法的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限和 IAM JSON 策略元素参考。

有关托管策略和内联策略之间的差别的更多信息，请参阅托管策略与内联策略。

Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

主题

- 删除 IAM 策略（控制台）
- 删除 IAM 策略（Amazon CLI）
- 删除 IAM 策略（Amazon API）

删除 IAM 策略（控制台）

您可以使用 Amazon Web Services 管理控制台 在 IAM 中删除客户管理型策略和内联策略。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

> **Note**
>
> 对 IAM policy 的删除是永久性的。策略删除后，无法恢复。

有关 IAM 策略结构和语法的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限和 IAM JSON 策略元素参考。

有关托管策略和内联策略之间的差别的更多信息，请参阅托管策略与内联策略。

先决条件

在删除策略之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

删除 IAM policy （控制台）

当客户管理型策略过时或不再符合您组织的安全要求和访问控制需求时，您可能需要将其删除。通过删除不必要的策略，可减少与过时或未使用的策略相关的潜在安全风险。您可删除客户管理型策略以便从您的 Amazon Web Services 账户 中删除它。您无法删除 Amazon 托管策略。

Console

删除客户管理型策略

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 选中您要删除的客户管理型策略旁边的复选框。您可以使用搜索框筛选策略列表。
4. 选择 Actions，然后选择 Delete。
5. 按照说明，确认您要删除该策略，然后选择删除。

删除内联策略（控制台）

当直接附加到内联策略的 IAM 用户、组或角色不再需要策略授予的特定权限时，您可能需要删除该策略。删除不必要的内联策略有助于降低意外访问的风险，尤其是内联策略无法像管理型策略那样在多个身份之间重复使用或共享。您可删除内联策略以便从您的 Amazon Web Services 账户 中删除它。您无法删除 Amazon 托管策略。

Console

删除 IAM 用户、组或角色的内联策略

1. 在导航窗格中，选择 Groups（组）、Users（用户）或 Roles（角色）。
2. 请选择具有要删除的策略的用户组、用户或角色的名称。然后选择 Permissions 选项卡。
3. 选中要删除的策略旁边的复选框，然后选择移除。然后，在确认对话框中，确认删除策略。

   - 要删除用户或角色中的内联策略，请选择移除以确认删除。
   - 如果您要删除 User groups（用户组）中的单个内联策略，请键入策略的名称，然后选择 Delete（删除）。如果要删除 User groups（用户组）中的多个内联策略，请键入要删除的策略数，接着键入 **`inline policies`**，然后选择 Delete（删除）。例如，如果要删除三个内联策略，请键入 **`3 inline policies`**。

删除 IAM 策略（Amazon CLI）

您可以使用 Amazon Command Line Interface（Amazon CLI）在 IAM 中删除客户管理型策略和内联策略。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

> **Note**
>
> 对 IAM policy 的删除是永久性的。策略删除后，无法恢复。

有关 IAM 策略结构和语法的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限和 IAM JSON 策略元素参考。

有关托管策略和内联策略之间的差别的更多信息，请参阅托管策略与内联策略。

先决条件

在删除策略之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

删除客户管理型策略（Amazon CLI）

您可以从 Amazon Command Line Interface 中删除客户托管策略。

删除客户托管策略 (Amazon CLI)

1. （可选）要查看有关策略的信息，请运行以下命令：
   - 列出托管策略：list-policies
   - 检索有关托管策略的详细信息：get-policy
2. （可选）要了解策略与身份之间的关系，请运行以下命令：
   - 要列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色），请运行以下命令：
     - list-entities-for-policy
   - 要列出附加到身份（用户、用户组或角色）的托管策略，请运行下列命令之一：
     - list-attached-user-policies
     - list-attached-group-policies
     - list-attached-role-policies
3. 要删除客户托管策略，请运行以下命令：
   - delete-policy

删除内联策略（Amazon CLI）

您可以从 Amazon CLI 中删除内联策略。

删除内联策略 (Amazon CLI)

1. （可选）要列出附加到身份（用户、用户组和角色）的所有内联策略，请使用以下命令之一：
   - aws iam list-user-policies
   - aws iam list-group-policies
   - aws iam list-role-policies

2. （可选）要检索嵌入到身份（用户、用户组或角色）中的内联策略文档，请使用以下命令之一：

   - aws iam get-user-policy
   - aws iam get-group-policy
   - aws iam get-role-policy

3. 要从身份（用户、用户组或不是服务相关角色的角色）中删除内联策略，请使用以下命令之一：

   - aws iam delete-user-policy
   - aws iam delete-group-policy
   - aws iam delete-role-policy

删除 IAM 策略（Amazon API）

您可以使用 Amazon API 在 IAM 中删除客户管理型策略和内联策略。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

> **Note**
>
> 对 IAM policy 的删除是永久性的。策略删除后，无法恢复。

有关 IAM 策略结构和语法的更多信息，请参阅 Amazon Identity and Access Management 中的策略和权限和 IAM JSON 策略元素参考。

有关托管策略和内联策略之间的差别的更多信息，请参阅托管策略与内联策略。

先决条件

在删除策略之前，您应查看其最近的服务级别活动。这非常重要，因为您不想删除使用它的主体（个人或应用程序）的访问权限。有关查看上次访问的信息的更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

删除客户管理型策略（Amazon API）

您可以使用 Amazon API 删除客户托管策略。

删除客户托管策略 (Amazon API)

1. （可选）要查看有关策略的信息，请调用以下操作：

- 列出托管策略：ListPolicies
- 检索有关托管策略的详细信息：GetPolicy

2. （可选）要了解策略与身份之间的关系，请调用以下操作：
   - 要列出托管式策略附加到的身份（IAM 用户、IAM 组和 IAM 角色），请调用以下操作：
     - ListEntitiesForPolicy
   - 要列出附加到身份（用户、用户组或角色）的托管策略，请调用以下一项操作：
     - ListAttachedUserPolicies
     - ListAttachedGroupPolicies
     - ListAttachedRolePolicies
3. 要删除客户托管策略，请调用以下操作：
   - DeletePolicy

删除 IAM 策略（Amazon API）

您可以使用 Amazon API 删除内联策略。

删除内联策略 (Amazon API)

1. （可选）要列出附加到身份（用户、用户组和角色）的所有内联策略，请调用以下操作之一：
   - ListUserPolicies
   - ListGroupPolicies
   - ListRolePolicies
2. （可选）要检索嵌入到身份（用户、用户组或角色）中的内联策略文档，请调用以下操作之一：
   - GetUserPolicy
   - GetGroupPolicy
   - GetRolePolicy
3. 要将内联策略从身份（用户、用户组或不是服务相关角色的角色）中删除，请调用以下操作之一：
   - DeleteUserPolicy
   - DeleteGroupPolicy
   - DeleteRolePolicy

使用上次访问的信息优化 Amazon 中的权限

作为管理员，您可能会向 IAM 资源（角色、用户、用户组或策略）授予超出需要的权限。IAM 提供上次访问的信息以帮助您识别未使用的权限，以便您可以将其移除。您可以使用上次访问信息来优化策略，并仅允许访问您的 IAM 身份和策略使用的服务和操作。这有助于更好地遵循最小权限的最佳实践。您可以查看 IAM 或 Amazon Organizations 中存在的身份或策略的上次访问信息。

您可以使用未使用的访问分析器持续监控上次访问的信息。有关更多信息，请参阅外部和未使用的访问的调查发现。

主题

- IAM 的上次访问的信息类型
- Amazon Organizations 的上次访问信息
- 关于上次访问的信息的知识
- 所需权限
- 排查 IAM 和 Amazon Organizations 实体的活动问题
- Amazon 跟踪上次访问信息的位置
- 查看 IAM 的上次访问信息
- 查看 Amazon Organizations 的上次访问信息
- 使用上次访问信息的示例应用场景
- 上次访问 IAM 操作信息的服务和操作

IAM 的上次访问的信息类型

对于 IAM 身份，您可以查看两种类型的上次访问信息：允许的 Amazon 服务信息和允许的操作信息。此信息包括尝试访问 Amazon 的日期和时间。对于操作，上次访问信息会报告服务管理操作。管理操作包括创建、删除和修改操作。要了解有关如何查看 IAM 的上次访问信息的更多信息，请参阅 查看 IAM 的上次访问信息。

有关在做有关向 IAM 身份授予权限的决策时使用上次访问信息的示例场景，请参阅 使用上次访问信息的示例应用场景。

要了解有关如何提供管理操作信息的更多信息，请参阅关于上次访问的信息的知识。

Amazon Organizations 的上次访问信息

如果使用管理账户凭证登录，则可以查看企业中的 Amazon Organizations 实体或策略的上次访问的服务信息。Amazon Organizations 实体包括企业根、企业部门 (OU) 或账户。Amazon Organizations 的上次访问信息包括有关服务控制策略 (SCP) 允许的服务的信息。该信息指示组织或账户中的哪些主体（根用户、IAM 用户或角色）上次尝试访问该服务以及尝试访问的时间。要了解有关报告以及如何查看 Amazon Organizations 的上次访问信息的更多信息，请参阅查看 Amazon Organizations 的上次访问信息。

有关使用上次访问信息做出有关向 Amazon Organizations 实体授予的权限的决策的示例场景，请参阅使用上次访问信息的示例应用场景。

关于上次访问的信息的知识

在使用报告中的上次访问信息更改 IAM 或 Amazon Organizations 实体的权限之前，请查看有关该信息的以下细节。

- 跟踪周期 - 最近的活动会在四小时内显示在 IAM 控制台中。服务信息的跟踪周期至少为 400 天，具体取决于服务何时开始跟踪操作信息。Amazon S3 操作信息的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM 和 Lambda 操作的跟踪周期从 2021 年 4 月 7 日开始。所有其他服务的跟踪周期从 2023 年 5 月 23 日开始计算。有关提供上次访问操作信息的服务列表，请参阅 上次访问 IAM 操作信息的服务和操作。要详细了解提供上次访问操作信息的区域，请参阅 Amazon 跟踪上次访问信息的位置。
- Attempts reported（报告的尝试）- 服务上次访问的数据包含访问 Amazon API 的所有尝试，而不仅仅是成功的尝试。这包括使用 Amazon Web Services 管理控制台、Amazon API（通过任何开发工具包）或任何命令行工具进行的所有尝试。在上次访问的服务相关数据中看到意外的条目并不意味着您的账户信息泄露，因为请求可能已被拒。请参阅您的 CloudTrail 日志并将其作为有关所有 API 调用以及它们是成功还是被拒绝的访问的信息的权威来源。
- PassRole - `iam:PassRole` 操作不会被跟踪，也不包括在 IAM 上次访问的服务信息中。
- 上次访问操作信息 – 对于 IAM 身份访问的服务管理操作，可提供上次访问操作信息。查看将报告上次访问操作信息的 服务与操作列表。

  > **Note**
  >
  > 对于任何数据平面事件，都没有上次访问操作信息。

- 管理事件 – IAM 将为由 CloudTrail 记录的服务管理事件提供操作信息。有时，CloudTrail 管理事件也称为控制层面操作或控制层面事件。通过管理事件，可以了解对在您 Amazon Web Services 账户

内的资源执行的管理操作。要详细了解 Cloudtrail 中的管理事件，请参阅《Amazon CloudTrail 用户指南》中的 记录管理事件。

- Report owner（报告拥有者）- 只有生成报告的主体才能查看报告详细信息。这意味着，当您查看 Amazon Web Services 管理控制台中的信息时，您可能需要等待它生成和加载。如果您使用 Amazon CLI 或 Amazon API 获取报告详细信息，则您的凭证必须与生成报告的主体的凭证相匹配。如果对角色或 Amazon STS 联合用户主体使用临时凭证，则必须在同一会话期间生成和检索报告。有关代入角色会话主体的更多信息，请参阅Amazon JSON 策略元素：Principal。
- IAM 资源 – IAM 的上次访问信息包括您账户中的 IAM 资源（角色、用户、IAM 组和策略）。Amazon Organizations 的上次访问信息包括指定 Amazon Organizations 实体中的主体（IAM 用户、IAM 角色或 Amazon Web Services 账户根用户）。上次访问信息不包括未通过身份验证的尝试信息。
- IAM policy 类型 - IAM 的上次访问信息包括 IAM 身份的策略允许的服务。这些是附加到角色或直接或通过组附加到用户的策略。其他策略类型允许的访问权限未包含在您的报告中。排除的策略类型包括基于资源的策略、访问控制列表、Amazon Organizations SCP、IAM 权限边界以及会话策略。由服务相关角色提供的权限由它们所关联的服务定义，不能在 IAM 中进行修改。要了解有关服务相关角色的更多信息，请参阅创建服务相关角色。要了解如何评估不同的策略类型以允许或拒绝访问，请参阅策略评估逻辑。
- Amazon Organizations 策略类型：Amazon Organizations 的信息仅包含 Amazon Organizations 实体继承的服务控制策略 (SCP) 允许的服务。SCP 是附加到根、OU 或账户的策略。其他策略类型允许的访问权限未包含在您的报告中。排除的策略类型包括基于身份的策略、基于资源的策略、访问控制列表、IAM 权限边界以及会话策略。要了解如何评估不同的策略类型以允许或拒绝访问，请参阅策略评估逻辑。
- 指定策略 ID：在 Amazon Organizations 中使用 Amazon CLI 或 Amazon API 为上次访问的信息生成报告时，您可以选择指定策略 ID。生成的报告包含仅该策略允许的服务的信息。该信息包括指定的 Amazon Organizations 实体或其子实体中的最新账户活动。有关更多信息，请参阅 aws iam generate-organizations-access-report 或 GenerateOrganizationsAccessReport。
- Amazon Organizations 管理账户：您必须登录到组织的管理账户才能查看服务的上次访问信息。您可以选择使用 IAM 控制台、Amazon CLI 或 Amazon API 查看管理账户的信息。生成的报告列出所有 Amazon 服务，因为管理账户不受 SCP 限制。如果在 CLI 或 API 中指定策略 ID，则会忽略该策略。对于每个服务，报告仅包含管理账户的信息。不过，其他 Amazon Organizations 实体的报告不会返回管理账户中的活动的信息。
- Amazon Organizations 设置：管理员必须先在组织根中启用 SCP，然后才能生成 Amazon Organizations 的数据。

所需权限

要在 Amazon Web Services 管理控制台中查看上次访问的信息，您必须具有授予所需权限的策略。

IAM 信息的权限

要使用 IAM 控制台查看 IAM 用户、角色或策略的上次访问信息，您必须具有包含以下操作的策略：

- `iam:GenerateServiceLastAccessedDetails`
- `iam:Get*`
- `iam:List*`

这些权限允许用户查看以下内容：

- 哪些用户、组或角色将附加到托管策略
- 用户或角色可访问哪些服务
- 他们上次访问服务的时间
- 他们上次尝试使用特定 Amazon EC2、IAM、Lambda 或 Amazon S3 操作的时间

要使用 Amazon CLI 或 Amazon API 查看 IAM 的上次访问信息，您必须具有与要使用的操作匹配的权限：

- `iam:GenerateServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetails`
- `iam:GetServiceLastAccessedDetailsWithEntities`
- `iam:ListPoliciesGrantingServiceAccess`

此示例说明如何创建基于身份的策略以允许查看 IAM 上次访问的信息。此外，它还允许对所有 IAM 进行只读访问。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
```

```
        "Action": [
            "iam:GenerateServiceLastAccessedDetails",
            "iam:Get*",
            "iam:List*"
        ],
        "Resource": "*"
    }
}
```

Amazon Organizations 信息的权限

要使用 IAM 控制台查看 Amazon Organizations 中的根、OU 或账户实体的报告，您必须具有包含以下操作的策略：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribeAccount
- organizations:DescribeOrganization
- organizations:DescribeOrganizationalUnit
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget
- organizations:ListRoots
- organizations:ListTargetsForPolicy

要使用 Amazon CLI 或 Amazon API 查看 Amazon Organizations 的上次访问的服务信息，您必须具有包含以下操作的策略：

- iam:GenerateOrganizationsAccessReport
- iam:GetOrganizationsAccessReport
- organizations:DescribePolicy
- organizations:ListChildren
- organizations:ListParents
- organizations:ListPoliciesForTarget

- `organizations:ListRoots`
- `organizations:ListTargetsForPolicy`

此示例说明如何创建基于身份的策略以允许查看 Amazon Organizations 的服务上次访问信息。此外，它还允许对所有 Amazon Organizations 进行只读访问。此策略定义了程序访问和控制台访问的权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": [
            "iam:GenerateOrganizationsAccessReport",
            "iam:GetOrganizationsAccessReport",
            "organizations:Describe*",
            "organizations:List*"
        ],
        "Resource": "*"
    }
}
```

您也可以使用 iam:OrganizationsPolicyId 条件键以仅允许为特定 Amazon Organizations 策略生成报告。有关策略示例，请参阅 IAM：查看 Amazon Organizations 策略的服务上次访问信息。

排查 IAM 和 Amazon Organizations 实体的活动问题

在某些情况下，Amazon Web Services 管理控制台上次访问的信息表可能为空。或者，Amazon CLI 或 Amazon API 请求可能会返回空信息集或空字段。在这些情况下，请检查以下问题：

- 对于上次访问的操作信息，您期望看到的操作可能不会在列表中返回。这可能是因为 IAM 身份不具有该操作的权限，或者 Amazon 尚未跟踪该操作的上次访问信息。
- 对于 IAM 用户，请确保该用户直接或通过组成员资格附加了至少一个内联或托管策略。
- 对于 IAM 组，请确认该组附加了至少一个内联或托管策略。
- 对于 IAM 组，该报告仅返回使用组的策略访问服务的成员的上次访问的服务信息。要了解成员是否使用了其他策略，请查看该用户的上次访问的信息。
- 对于 IAM 角色，请确认该角色附加了至少一个内联或托管策略。

- 对于 IAM 实体（用户或角色），请检查可能影响该实体的权限的其他策略类型。其中包含基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 策略类型 或 对单个账户内的请求进行策略评估。
- 对于 IAM policy，请确保指定的托管策略附加到至少一个用户、具有成员的组或角色。
- 对于 Amazon Organizations 实体（根、OU 或账户），请确保您使用 Amazon Organizations 管理账户凭证登录。
- 确认在组织根中启用了 SCP。
- 上次访问的操作信息仅适用于 上次访问 IAM 操作信息的服务和操作 中列出的操作。

在进行更改时，请等待至少四小时，以便在 IAM 控制台报告中显示活动。如果您使用 Amazon CLI 或 Amazon API，则必须生成新的报告以查看更新后的信息。

Amazon 跟踪上次访问信息的位置

Amazon 收集标准 Amazon 区域的上次访问信息。在 Amazon 添加额外的区域时，这些区域将添加到下表中，包括 Amazon 开始在每个区域中跟踪信息的日期：

- 服务信息 - 服务的跟踪周期至少为 400 天，如果您的区域在过去 400 天内才开始跟踪此功能，则此周期会更短。
- 操作信息 - Amazon S3 管理操作的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM 和 Lambda 管理操作的跟踪周期从 2021 年 4 月 7 日开始。所有其他服务的管理操作跟踪周期从 2023 年 5 月 23 日开始计算。如果某个区域的开始跟踪日期为 2023 年 5 月 23 日以后，则该区域的上次操作访问信息将从该较晚的日期开始。

区域名称	区域	跟踪开始日期
美国东部（俄亥俄州）	us-east-2	2017 年 10 月 27 日
美国东部（弗吉尼亚州北部）	us-east-1	2015 年 10 月 1 日
美国西部（北加利福尼亚）	us-west-1	2015 年 10 月 1 日
美国西部（俄勒冈州）	us-west-2	2015 年 10 月 1 日
非洲（开普敦）	af-south-1	2020 年 4 月 22 日
亚太地区（香港）	ap-east-1	2019 年 4 月 24 日

区域名称	区域	跟踪开始日期
亚太地区（海得拉巴）	ap-south-2	2022 年 11 月 22 日
亚太地区（雅加达）	ap-southeast-3	2021 年 12 月 13 日
亚太地区（墨尔本）	ap-southeast-4	2023 年 1 月 23 日
亚太地区（孟买）	ap-south-1	2016 年 6 月 27 日
亚太地区（大阪）	ap-northeast-3	2018 年 2 月 11 日
亚太地区（首尔）	ap-northeast-2	2016 年 1 月 6 日
亚太地区（新加坡）	ap-southeast-1	2015 年 10 月 1 日
亚太地区（悉尼）	ap-southeast-2	2015 年 10 月 1 日
亚太地区（东京）	ap-northeast-1	2015 年 10 月 1 日
加拿大（中部）	ca-central-1	2017 年 10 月 28 日
欧洲地区（法兰克福）	eu-central-1	2015 年 10 月 1 日
欧洲地区（爱尔兰）	eu-west-1	2015 年 10 月 1 日
欧洲（伦敦）	eu-west-2	2017 年 10 月 28 日
欧洲地区（米兰）	eu-south-1	2020 年 4 月 28 日
欧洲地区（巴黎）	eu-west-3	2017 年 12 月 18 日
欧洲（西班牙）	eu-south-2	2022 年 11 月 15 日
欧洲地区（斯德哥尔摩）	eu-north-1	2018 年 12 月 12 日
欧洲（苏黎世）	eu-central-2	2022 年 11 月 8 日
以色列（特拉维夫）	il-central-1	2023 年 8 月 1 日
中东（巴林）	me-south-1	2019 年 7 月 29 日

区域名称	区域	跟踪开始日期
中东（阿联酋）	me-central-1	2022 年 8 月 30 日
南美洲（圣保罗）	sa-east-1	2015 年 12 月 11 日
Amazon GovCloud（美国东部）	us-gov-east-1	2023 年 7 月 1 日
Amazon GovCloud（美国西部）	us-gov-west-1	2023 年 7 月 1 日

如果某个区域未在上表中列出，则表明此区域尚不提供上次访问的信息。

Amazon 区域是地理区域中的 Amazon 资源集合。区域分组为分区。标准区域是属于 aws 分区的区域。有关不同分区的更多信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 资源名称 (ARN) 格式。有关区域的更多信息，请参阅 Amazon Web Services 一般参考 中的关于 Amazon 区域。

查看 IAM 的上次访问信息

您可以使用 Amazon Web Services 管理控制台、Amazon CLI 或 Amazon API 查看 IAM 的上次访问信息。查看将显示上次访问信息的 服务与操作列表。有关上次访问信息的更多信息，请参阅使用上次访问的信息优化 Amazon 中的权限。

您可以在 IAM 中查看下列资源类型的信息。在每种情况下，该信息包括给定报告周期允许的服务：

- 用户 - 查看有关用户上次尝试访问每个允许的服务的信息。
- User group（用户组）- 查看有关用户组成员上次尝试访问每个允许的服务的信息。此报告还包括已尝试访问的成员的总数。
- Role（角色）- 查看有关某个人上次使用角色尝试访问每个允许的服务的信息。
- Policy（策略）- 查看有关用户或角色上次尝试访问每个允许的服务的信息。此报告还包括已尝试访问的实体的总数。

> **Note**
>
> 在 IAM 中查看资源的访问信息之前，请确保您了解信息的报告周期、报告的实体和已评估的策略类型。有关更多详细信息，请参阅the section called “关于上次访问的信息的知识”。

查看 IAM（控制台）的信息

您可以在 IAM 控制台的上次访问选项卡上查看 IAM 的上次访问信息。

查看 IAM（控制台）的信息

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择 Groups（组）、Users（用户）、Roles（角色）或 Policies（策略）。
3. 请选择任意用户、用户组、角色或策略名称打开相应的摘要页面，然后选择上次访问时间选项卡。根据您选择的资源查看以下信息：
   - User group（用户组）- 查看用户组成员（用户）可以访问的服务列表。您还可以查看成员上次访问服务的时间、成员使用的用户组策略以及发出请求的用户组成员。请选择策略的名称以了解它是托管策略还是内联用户组策略。请选择用户组成员的名称以查看该用户组的所有成员以及他们上次访问服务的时间。
   - User（用户）- 查看用户可以访问的服务列表。您还可以查看这些用户和角色上次访问服务的时间，以及当前与该用户关联的策略。请选择策略的名称以了解它是托管策略、内联用户策略还是用户组的内联策略。
   - Role（角色）- 查看角色可以访问的服务的列表、角色上次访问服务的时间以及所使用的策略。选择策略的名称以了解它是托管策略还是内联角色策略。
   - Policy（策略）- 查看策略中允许的操作的服务列表。还可以查看上次使用策略访问服务的时间以及使用该策略的实体（用户或角色）。上次访问日期还包括通过其他策略授予对此策略的访问权限的时间。选择实体的名称以了解哪些实体已附加此策略以及实体上次访问服务的时间。
4. 在表中的服务列中，选择 包含上次访问操作信息的一种服务 的名称，以查看 IAM 实体尝试访问的管理操作的列表。您可以查看 Amazon Web Services 区域以及时间戳（显示某个人上次尝试执行操作的时间）。
5. 对于 包含上次访问操作信息的服务 的服务和管理操作，将会显示上次访问时间列。查看此列中返回的以下可能结果。这些结果会有所不同，具体取决于是否允许某个服务或操作、是否访问了此服务或操作，以及 Amazon 是否跟踪此服务或操作以获取上次访问的信息。

<number of> 天前

自跟踪周期内使用服务或操作以来的天数。服务的跟踪周期为过去 400 天。Amazon S3 操作的跟踪周期从 2020 年 4 月 12 日开始。Amazon EC2、IAM Lambda 和操作的跟踪周期从 2021 年 4 月 7 日开始。所有其他服务的跟踪周期从 2023 年 5 月 23 日开始计算。要详细了解每个 Amazon Web Services 区域的跟踪开始日期，请参阅 Amazon 跟踪上次访问信息的位置。

在跟踪周期间未访问

所跟踪的服务或操作在跟踪周期内未被实体使用。

您可能对未出现在列表中的操作拥有权限。如果 Amazon 当前未包含操作的跟踪信息，则可能会发生这种情况。您不应仅出于缺少跟踪信息来做出权限决定。相反，我们建议您使用此信息来告知和支持授予最小权限的总体策略。检查您的策略以确认访问级别是否适当。

查看 IAM 的信息 (Amazon CLI)

您可以使用 Amazon CLI 检索有关上次使用 IAM 资源尝试访问 Amazon 服务和 Amazon S3、Amazon EC2、IAM 以及 Lambda 操作的信息。IAM 资源可能是用户、用户组、角色或策略。

查看 IAM 的信息 (Amazon CLII)

1. 生成报告。请求必须包括要报告的 IAM 资源（用户、用户组、角色或策略）的 ARN。您可以指定要在报告中生成的粒度级别，以查看服务或服务和操作的访问详细信息。该请求返回一个 `job-id`，您之后可在 `get-service-last-accessed-details` 和 `get-service-last-accessed-details-with-entities` 操作中使用它来监控 `job-status`，直到作业完成。

   - aws iam generate-service-last-accessed-details

2. 检索有关使用上一步中的 `job-id` 参数的报告的详细信息。

   - aws iam get-service-last-accessed-details

   此操作根据您在 `generate-service-last-accessed-details` 操作中请求的资源类型和粒度级别返回以下信息：

   - User（用户）- 返回指定用户可访问的服务的列表。对于每个服务，此操作返回用户上次尝试的日期和时间以及用户的 ARN。

- 用户组 — 返回指定用户组的成员可使用附加到用户组的策略访问的服务列表。对于每个服务，此操作返回任何用户组成员（用户）上次尝试的日期和时间。它还返回该用户的 ARN 以及已尝试访问服务的用户组成员的总数。使用 GetServiceLastAccessedDetailsWithEntities 操作可检索所有成员的列表。
- Role（角色）- 返回指定角色可访问的服务的列表。对于每个服务，此操作返回角色上次尝试的日期和时间以及角色的 ARN。
- Policy（策略）- 返回指定策略允许访问的服务的列表。对于每个服务，此操作返回实体（用户或角色）上次尝试使用策略访问服务的日期和时间。它还返回实体的 ARN 以及已尝试访问的实体的总数。

3. 了解有关在尝试访问特定服务时使用用户组或策略权限的实体的更多信息。此操作返回具有每个实体的 ARN、ID、名称、路径、类型（用户或角色）的实体列表以及实体上次尝试访问服务的时间。您还可以对用户和角色使用此操作，但它仅返回有关该实体的信息。

   - aws iam get-service-last-accessed-details-with-entities

4. 了解有关在尝试访问特定服务时身份（用户、用户组或角色）使用的基于身份的策略的更多信息。在指定身份和服务时，此操作返回身份可用于访问指定服务的权限策略的列表。此操作提供策略的当前状态，而不依赖于生成的报告。它也不返回其他策略类型，例如基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 策略类型 或 对单个账户内的请求进行策略评估。

   - aws iam list-policies-granting-service-access

查看 IAM 的信息 (Amazon API)

您可以使用 Amazon API 检索有关上次使用 IAM 资源尝试访问 Amazon 服务和 Amazon S3、Amazon EC2、IAM 以及 Lambda 操作的信息。IAM 资源可能是用户、用户组、角色或策略。您可以指定要在报告中生成的粒度级别，以查看服务或服务和操作的详细信息。

查看 IAM 的信息 (Amazon API)

1. 生成报告。请求必须包括要报告的 IAM 资源（用户、用户组、角色或策略）的 ARN。它返回一个 `JobId`，您之后可在 `GetServiceLastAccessedDetails` 和 `GetServiceLastAccessedDetailsWithEntities` 操作中使用它来监控 `JobStatus`，直到作业完成。

   - GenerateServiceLastAccessedDetails

2. 检索有关使用上一步中的 `JobId` 参数的报告的详细信息。

- GetServiceLastAccessedDetails

此操作根据您在 `GenerateServiceLastAccessedDetails` 操作中请求的资源类型和粒度级别返回以下信息：

- User（用户）- 返回指定用户可访问的服务的列表。对于每个服务，此操作返回用户上次尝试的日期和时间以及用户的 ARN。
- 用户组 — 返回指定用户组的成员可使用附加到用户组的策略访问的服务列表。对于每个服务，此操作返回任何用户组成员（用户）上次尝试的日期和时间。它还返回该用户的 ARN 以及已尝试访问服务的用户组成员的总数。使用 GetServiceLastAccessedDetailsWithEntities 操作可检索所有成员的列表。
- Role（角色）- 返回指定角色可访问的服务的列表。对于每个服务，此操作返回角色上次尝试的日期和时间以及角色的 ARN。
- Policy（策略）- 返回指定策略允许访问的服务的列表。对于每个服务，此操作返回实体（用户或角色）上次尝试使用策略访问服务的日期和时间。它还返回实体的 ARN 以及已尝试访问的实体的总数。

3. 了解有关在尝试访问特定服务时使用用户组或策略权限的实体的更多信息。此操作返回具有每个实体的 ARN、ID、名称、路径、类型（用户或角色）的实体列表以及实体上次尝试访问服务的时间。您还可以对用户和角色使用此操作，但它仅返回有关该实体的信息。

   - GetServiceLastAccessedDetailsWithEntities

4. 了解有关在尝试访问特定服务时身份（用户、用户组或角色）使用的基于身份的策略的更多信息。在指定身份和服务时，此操作返回身份可用于访问指定服务的权限策略的列表。此操作提供策略的当前状态，而不依赖于生成的报告。它也不返回其他策略类型，例如基于资源的策略、访问控制列表、Amazon Organizations 策略、IAM 权限边界或会话策略。有关更多信息，请参阅 策略类型 或 对单个账户内的请求进行策略评估。

   - ListPoliciesGrantingServiceAccess

查看 Amazon Organizations 的上次访问信息

您可以使用 IAM 控制台、Amazon CLI 或 Amazon API 查看 Amazon Organizations 的上次访问的服务信息。有关数据、所需的权限、故障排除和支持的区域的重要信息，请参阅使用上次访问的信息优化 Amazon 中的权限。

在使用 Amazon Organizations 管理账户凭证登录到 IAM 控制台时，您可以查看组织中的任何实体的信息。Amazon Organizations 实体包括组织根、组织部门 (OU) 和账户。您还可以使用 IAM 控制台查看企业中的任何服务控制策略 (SCP) 的信息。IAM 显示适用于实体的任何 SCP 允许的服务列表。对于每个服务，您可以查看所选的 Amazon Organizations 实体或其子实体的最新账户活动信息。

在将 Amazon CLI 或 Amazon API 与管理账户凭证一起使用时，您可以为企业中的任何实体或策略生成报告。实体的编程报告包括适用于该实体的任何 SCP 允许的服务列表。对于每个服务，该报告包括指定的 Amazon Organizations 实体或其子树中的账户的最新活动。

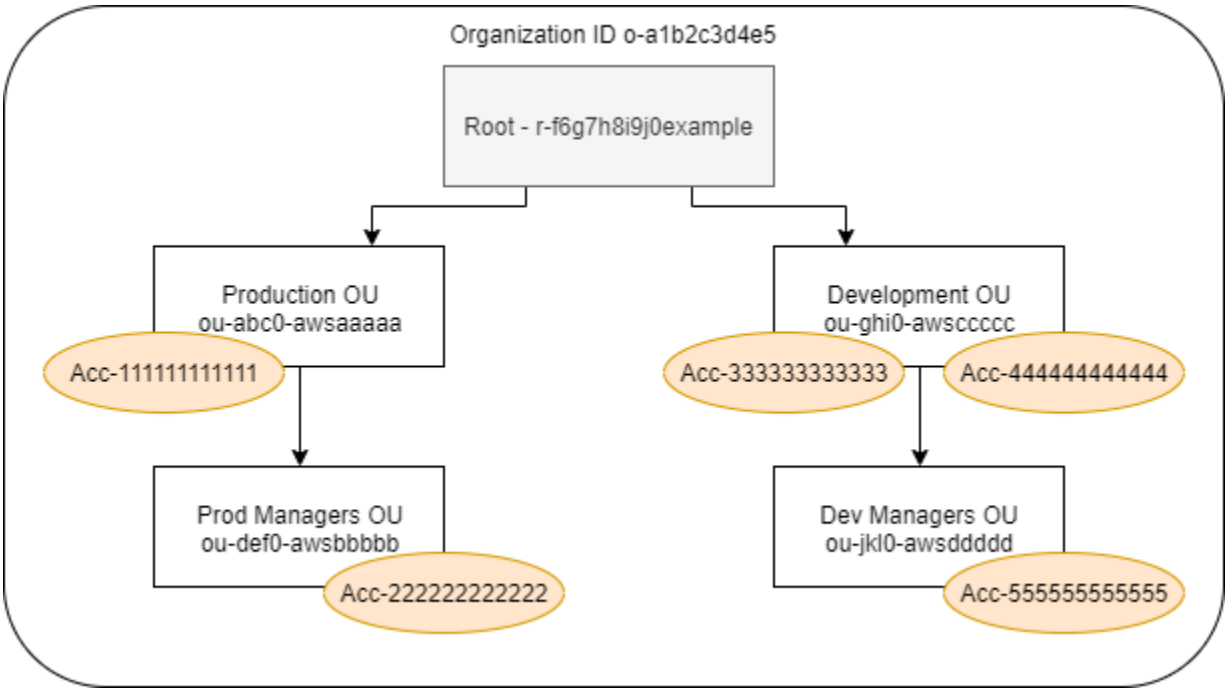
在为策略生成编程报告时，您必须指定一个 Amazon Organizations 实体。该报告包括指定的 SCP 允许的服务列表。对于每个服务，它包括通过该策略授予权限的实体或其子实体中的最新账户活动。有关更多信息，请参阅 aws iam generate-organizations-access-report 或 GenerateOrganizationsAccessReport。

在查看报告之前，请确保您了解管理账户要求和信息、报告周期、报告的实体和评估的策略类型。有关更多详细信息，请参阅the section called “关于上次访问的信息的知识”。

了解 Amazon Organizations 实体路径

使用 Amazon CLI 或 Amazon API 生成 Amazon Organizations 访问报告时，必须指定实体路径。路径是 Amazon Organizations 实体结构的文本表示形式。

您可以使用组织的已知结构构建实体路径。例如，假定您在 Amazon Organizations 中有以下组织结构。



Dev Managers OU 的路径是使用组织 ID、根目录及该路径下的所有 OU（包括 Dev Managers OU）构建的。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-ghi0-awsccccc/ou-jkl0-awsddddd/
```

Production OU 中账户的路径是使用组织 ID、根目录、该 OU 和账号构建的。

```
o-a1b2c3d4e5/r-f6g7h8i9j0example/ou-abc0-awsaaaaa/111111111111/
```

> **Note**
>
> 组织 ID 是全局唯一的，但 OU ID 和根目录 ID 仅在组织内是唯一的。这意味着没有两个组织具有相同的组织 ID。但是，另一个组织可能具有与您的组织相同的 OU 或根目录 ID。我们建议您在指定 OU 或根目录时始终包含组织 ID。

查看 Amazon Organizations 的信息（控制台）

您可以使用 IAM 控制台查看根、OU、账户或策略的上次访问的服务信息。

查看根目录的信息（控制台）

1. 使用 Amazon Organizations 管理账户凭证登录到Amazon Web Services 管理控制台，然后打开 IAM 控制台，地址：https://console.aws.amazon.com/iam/。
2. 在导航窗格中的 Access reports (访问报告) 部分下，选择 Organization activity (组织活动)。
3. 在 Organization activity (组织活动) 页面上，选择 Root (根)。
4. 在 Details and activity (详细信息和活动) 选项卡上，查看 Service access report (服务访问报告) 部分。该信息包括直接附加到根的策略允许的服务列表。该信息显示您上次从中访问服务的账户以及访问时间。有关访问服务的主体的更多详细信息，请以该账户中的管理员身份登录并查看 IAM 服务上次访问的服务信息。
5. 请选择 Attached SCPs（附加的 SCP）选项卡以查看附加到根的服务控制策略 (SCP）列表。IAM 显示您可以查看每个策略附加到的目标实体数量。您可以使用该信息确定要检查的 SCP。
6. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务，查看上次从中访问该服务的账户以及访问时间。
7. 选择在 Amazon Organizations 中编辑，以查看其他详细信息并在 Amazon Organizations 控制台中编辑 SCP。有关更多信息，请参阅 Amazon Organizations 用户指南中的更新 SCP。

查看 OU 或账户的信息（控制台）

1. 使用 Amazon Organizations 管理账户凭证登录到Amazon Web Services 管理控制台，然后打开 IAM 控制台，地址：https://console.aws.amazon.com/iam/。
2. 在导航窗格中的 Access reports (访问报告) 部分下，选择 Organization activity (组织活动)。
3. 在 Organization activity (组织活动) 页面上，展开您的组织的结构。接下来，请选择要查看的 OU 或任何账户的名称，但管理账户除外。
4. 在 Details and activity (详细信息和活动) 选项卡上，查看 Service access report (服务访问报告) 部分。该信息包括附加到 OU 或账户及其 所有父项的 SCP 允许的服务列表。该信息显示您上次从中访问服务的账户以及访问时间。有关访问服务的主体的更多详细信息，请以该账户中的管理员身份登录并查看 IAM 服务上次访问的服务信息。
5. 请选择 Attached SCPs（附加的 SCP）选项卡以查看直接附加到 OU 或账户的服务控制策略 (SCP) 列表。IAM 显示您可以查看每个策略附加到的目标实体数量。您可以使用该信息确定要检查的 SCP。
6. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务，查看上次从中访问该服务的账户以及访问时间。
7. 选择在 Amazon Organizations 中编辑，以查看其他详细信息并在 Amazon Organizations 控制台中编辑 SCP。有关更多信息，请参阅 Amazon Organizations 用户指南中的更新 SCP。

查看管理账户的信息（控制台）

1. 使用 Amazon Organizations 管理账户凭证登录到Amazon Web Services 管理控制台，然后打开 IAM 控制台，地址：https://console.aws.amazon.com/iam/。
2. 在导航窗格中的 Access reports (访问报告) 部分下，选择 Organization activity (组织活动)。
3. 在 Organization activity（企业活动）页面上，展开您的企业的结构，然后选择管理账户的名称。
4. 在 Details and activity (详细信息和活动) 选项卡上，查看 Service access report (服务访问报告) 部分。该信息包括所有 Amazon 服务的列表。管理账户不受 SCP 限制。该信息显示账户上次是否访问服务以及访问时间。有关访问服务的主体的更多详细信息，请以该账户中的管理员身份登录并查看 IAM 服务上次访问的服务信息。
5. 请选择 Attached SCPs（附加的 SCP）选项卡以确认没有附加的 SCP，因为该账户是管理账户。

查看策略的信息（控制台）

1. 使用 Amazon Organizations 管理账户凭证登录到Amazon Web Services 管理控制台，然后打开 IAM 控制台，地址：https://console.aws.amazon.com/iam/。

2. 在导航窗格中的 Access reports (访问报告) 部分下，选择 Service control policies (SCPs) (服务控制策略(SCP))。
3. 在 Service control policies (SCPs) (服务控制策略 (SCP)) 页面上，查看您组织中的策略列表。您可以查看每个策略附加到的目标实体数量。
4. 请选择一个 SCP 名称以查看该策略允许的所有服务。对于每个服务，查看上次从中访问该服务的账户以及访问时间。
5. 选择在 Amazon Organizations 中编辑，以查看其他详细信息并在 Amazon Organizations 控制台中编辑 SCP。有关更多信息，请参阅 Amazon Organizations 用户指南中的更新 SCP。

查看 Amazon Organizations 的信息 (Amazon CLI)

您可以使用 Amazon CLI 检索 Amazon Organizations 根、OU、账户或策略的上次访问的服务信息。

查看 Amazon Organizations 上次访问的服务信息 (Amazon CLI)

1. 使用具有所需的 IAM 和 Amazon Organizations 权限的 Amazon Organizations 管理账户凭证，并确认为根启用了 SCP。有关更多信息，请参阅 关于上次访问的信息的知识。
2. 生成报告。请求必须包含要生成报告的 Amazon Organizations 实体（根、OU 或账户）的路径。您可以选择包含 `organization-policy-id` 参数以查看特定策略的报告。该命令返回 `job-id`，您可以随后在 `get-organizations-access-report` 命令中使用该内容以监控 `job-status`，直到作业完成。

   - aws iam generate-organizations-access-report
3. 检索有关使用上一步中的 `job-id` 参数的报告的详细信息。

   - aws iam get-organizations-access-report

   该命令返回实体成员可以访问的服务列表。对于每个服务，该命令返回账户成员的上次尝试日期和时间以及账户的实体路径。它还返回可访问的服务总数以及未访问的服务数。如果指定了可选的 `organizations-policy-id` 参数，则可访问的服务是指定策略允许的那些服务。

查看 Amazon Organizations 的信息 (Amazon API)

您可以使用 Amazon API 检索 Amazon Organizations 根、OU、账户或策略的上次访问的服务信息。

查看 Amazon Organizations 上次访问的服务信息 (Amazon API)

1. 使用具有所需的 IAM 和 Amazon Organizations 权限的 Amazon Organizations 管理账户凭证，并确认为根启用了 SCP。有关更多信息，请参阅 关于上次访问的信息的知识。
2. 生成报告。请求必须包含要生成报告的 Amazon Organizations 实体（根、OU 或账户）的路径。您可以选择包含 `OrganizationsPolicyId` 参数以查看特定策略的报告。该操作返回 `JobId`，您可以随后在 `GetOrganizationsAccessReport` 操作中使用该内容以监控 `JobStatus`，直到作业完成。

   - GenerateOrganizationsAccessReport
3. 检索有关使用上一步中的 `JobId` 参数的报告的详细信息。

   - GetOrganizationsAccessReport

   该操作返回实体成员可以访问的服务列表。对于每个服务，该操作返回账户成员的上次尝试日期和时间以及账户的实体路径。它还返回可访问的服务总数以及未访问的服务数。如果指定了可选的 `OrganizationsPolicyId` 参数，则可访问的服务是指定策略允许的那些服务。

使用上次访问信息的示例应用场景

您可以使用上次访问的信息确定您为 IAM 实体或 Amazon Organizations 实体授予的权限。有关更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

> **Note**
>
> 在 IAM 或 Amazon Organizations 中查看实体或策略的访问信息之前，请确保您了解信息的报告周期、报告的实体和评估的策略类型。有关更多详细信息，请参阅the section called “关于上次访问的信息的知识”。

作为管理员，您可以平衡适合您的公司的可访问性和最小权限。

使用信息减少 IAM 组的权限

您可以使用上次访问的信息减少 IAM 组权限，以仅包含用户所需的服务。此方法是在服务级别授予最小权限的重要步骤。

例如，Paulo Santos 是负责为 Example Corp 定义 Amazon 用户权限的管理员。这家公司刚刚开始使用 Amazon，而软件开发团队尚未定义他们将使用何种 Amazon 服务。Paulo 希望仅向团队授予访问他

们所需的服务的权限，但由于尚未定义，因此他暂时向团队授予高级用户权限。然后，他使用上次访问的信息来减少组的权限。

Paulo 使用以下 JSON 文本创建名为 `ExampleDevelopment` 的托管策略。然后，他将该策略附加到名为 `Development` 的组，并将所有开发人员添加到该组。

> **Note**
>
> Paulo 的高级用户可能需要 `iam:CreateServiceLinkedRole` 权限才能使用某些服务和功能。他知道添加此权限将允许用户创建任何服务相关角色。他为高级用户接受这种风险。

JSON

```
{

    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToAllServicesExceptPeopleManagement",
            "Effect": "Allow",
            "NotAction": [
                "iam:*",
                "organizations:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
    ]
}
```

Paulo 决定等待 90 天，之后再使用 Amazon Web Services 管理控制台查看上次访问的信息（适用于 Development 组）。他查看组成员访问过的服务的列表。他了解到，用户上周访问了 5 项服务：Amazon CloudTrail、Amazon CloudWatch Logs、Amazon EC2、Amazon KMS 和 Amazon S3.. 他们在首次评估 Amazon 时访问过其他几项服务，但之后便未访问。

Paulo 决定减少策略权限以仅包含上述 5 项服务以及所需的 IAM 和 Amazon Organizations 操作。他使用以下 JSON 文本编辑 ExampleDevelopment 策略。

> **Note**
>
> Paulo 的高级用户可能需要 iam:CreateServiceLinkedRole 权限才能使用某些服务和功能。他知道添加此权限将允许用户创建任何服务相关角色。他为高级用户接受这种风险。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccessToListedServices",
            "Effect": "Allow",
            "Action": [
                "s3:*",
                "kms:*",
                "cloudtrail:*",
                "logs:*",
                "ec2:*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "RequiredIamAndOrgsActions",
            "Effect": "Allow",
            "Action": [
                "iam:CreateServiceLinkedRole",
                "iam:ListRoles",
                "organizations:DescribeOrganization"
            ],
            "Resource": "*"
        }
```

```
    ]
}
```

为了进一步减少权限，Paulo 可在 Amazon CloudTrail Event history (事件历史记录) 中查看账户的事件。在那里，他可以查看可用于减少策略权限的详细事件信息，以仅包括开发人员所需的操作和资源。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的在 CloudTrail 控制台中查看 CloudTrail 事件。

使用信息减少 IAM 用户的权限

您可以使用上次访问的信息来减少单个 IAM 用户的权限。

例如，Martha Rivera 是 IT 管理员，她负责确保公司的人员没有多余的 Amazon 权限。在定期安检中，她检查所有 IAM 用户的权限。这些用户中，有一个是名叫 Nikhil Jayashankar 的应用程序开发人员，以前担任过安全工程师。由于工作要求发生变化，Nikhil 成为了 `app-dev` 组和 `security-team` 组的成员。适用于他的新工作的 `app-dev` 组授予了对多项服务（包括 Amazon EC2、Amazon EBS、Auto Scaling、Amazon S3、Route 53 和 Elastic Transcoder）的权限。他的旧工作的 `security-team` 组授予了对 IAM 和 CloudTrail 的权限。

Martha 以管理员身份登录 IAM 控制台，并依次选择用户、姓名 `nikhilj` 和上次访问时间选项卡。

Martha 查看 Last Accessed（上次访问时间）列，发现 Nikhil 最近未访问过 IAM、CloudTrail、Route 53、Amazon Elastic Transcoder 以及很多其他 Amazon 服务。Nikhil 已访问 Amazon S3。Martha 从服务列表中选择 S3，并了解到 Nikhil 在过去两周内执行了部分 Amazon S3 `List` 操作。在其公司内部，Martha 确认 Nikhil 不再有访问 IAM 和 CloudTrail 的业务需要，因为他不再是内部安全团队的成员。

Martha 管理员现在准备好处理上次访问的服务和操作信息。但是，与上一个示例中的组不同，像 `nikhilj` 这样的 IAM 用户可能受多个策略的约束并且是多个组的成员。Martha 必须谨慎行事，以避免无意中破坏 `nikhilj` 或其他组成员的访问权限。除了了解 Nikhil 应拥有的访问权限之外，她还必须确定他如何接收这些权限。

Martha 选择 Permissions (权限) 选项卡，可在其中查看直接附加到 `nikhilj` 的策略以及从组附加的策略。她扩展每个策略并查看策略摘要，以了解哪个策略允许访问 Nikhil 未使用的服务：

- IAM – `IAMFullAccess` Amazon 托管策略直接附加到 `nikhilj` 并附加到 `security-team` 组。
- CloudTrail - `AmazonCloudTrailReadOnlyAccess` Amazon 托管策略附加到 `security-team` 组。
- Route 53 – `App-Dev-Route53` 客户托管策略附加到 `app-dev` 组。

- Elastic Transcoder - App-Dev-ElasticTranscoder 客户托管策略附加到 app-dev 组。

Martha 决定删除直接附加到 nikhilj 的 IAMFullAccess Amazon 托管策略。她还删除 Nikhil 的 security-team 组成员资格。这两项操作删除了对 IAM 和 CloudTrail 的不必要的访问权限。

Nikhil 对 Route 53 和 Elastic Transcoder 的访问权限由 app-dev 组授予。虽然 Nikhil 未使用这些服务，但组的其他成员可能使用了这些服务。Martha 查看 app-dev 组的上次访问信息，并了解到有几个成员最近访问了 Route 53 和 Amazon S3。但在过去一年中，没有任何组成员访问过 Elastic Transcoder。她从组中删除 App-Dev-ElasticTranscoder 客户托管策略。

然后，Martha 查看 App-Dev-ElasticTranscoder 客户托管策略的上次访问的信息。她了解到该策略未附加到任何其他 IAM 身份。她在公司内部进行调查，以确保将来不再需要该策略，然后将其删除。

在删除 IAM 资源之前使用信息

在删除 IAM 资源之前，您可以使用上次访问的信息，以确保自上次使用该资源以来已经过了一定的时间。这适用于用户、组、角色和策略。要了解这些操作的更多信息，请参阅以下主题：

- IAM 用户 – 移除或停用 IAM 用户
- 群组 – 删除 IAM 群组
- 角色 – 删除角色或实例配置文件
- 策略 - 删除 IAM 策略（这还会将策略与身份分离）

在编辑 IAM policy 之前使用信息

您可以先查看 IAM 身份（用户、组或角色）或 IAM policy 的上次访问信息，然后再编辑影响该资源的策略。这很重要，因为您不想删除正在使用它的人员的访问权限。

例如，Arnav Desai 是 Example Corp 的开发人员和 Amazon 管理员。当他的团队开始使用 Amazon，他们为所有开发人员提供了高级用户访问权限，允许他们完全访问除 IAM 和 Amazon Organizations 之外的所有服务。作为授予最小权限操作的第一步，Arnav 希望使用 Amazon CLI 来查看其账户中的托管策略。

为此，Arnav 首先使用以下命令在其账户中列出附加到身份的客户托管权限策略：

```
aws iam list-policies --scope Local --only-attached --policy-usage-filter
 PermissionsPolicy
```

从响应中，他捕获了每个策略的 ARN。然后，Arnav 使用以下命令为每个策略生成上次访问的信息报告。

```
aws iam generate-service-last-accessed-details --arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

在该响应中，他从 JobId 字段中捕获了生成的报告的 ID。然后，Arnav 轮询以下命令，直到 JobStatus 字段返回 COMPLETED 或 FAILED 值。如果作业失败，他将捕获错误。

```
aws iam get-service-last-accessed-details --job-id 98a765b4-3cde-2101-2345-example678f9
```

当作业的状态为 COMPLETED 时，Arnav 将解析 JSON 格式的 ServicesLastAccessed 数组的内容。

```
 "ServicesLastAccessed": [
        {
            "TotalAuthenticatedEntities": 1,
            "LastAuthenticated": 2018-11-01T21:24:33.222Z,
            "ServiceNamespace": "dynamodb",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/IAMExampleUser",
            "ServiceName": "Amazon DynamoDB"
        },

        {
            "TotalAuthenticatedEntities": 0,
            "ServiceNamespace": "ec2",
            "ServiceName": "Amazon EC2"
        },

        {
            "TotalAuthenticatedEntities": 3,
            "LastAuthenticated": 2018-08-25T15:29:51.156Z,
            "ServiceNamespace": "s3",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:role/IAMExampleRole",
            "ServiceName": "Amazon S3"
        }
    ]
```

通过此信息，Arnav 了解到 ExamplePolicy1 策略允许访问三项服务，即 Amazon DynamoDB、Amazon S3 和 Amazon EC2。名为 IAMExampleUser 的 IAM 用户上次于 11 月 1 日尝

试访问过 DynamoDB，并且有人已于 8 月 25 日使用 IAMExampleRole 角色尝试访问 Amazon S3。此外，还有另外两个实体在去年尝试访问过 Amazon S3。但是，去年没有人尝试访问 Amazon EC2。

这意味着，Arnav 可从策略中安全地删除 Amazon EC2 操作。Arnav 想查看策略的当前 JSON 文档。首先，他必须使用以下命令确定策略的版本号。

```
aws iam list-policy-versions --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1
```

从响应中，Arnav 收集 Versions 数组中的当前默认版本号。然后，他使用该版本号 (v2) 通过以下命令请求 JSON 策略文档。

```
aws iam get-policy-version --policy-arn arn:aws:iam::123456789012:policy/ExamplePolicy1
 --version-id v2
```

Arnav 存储 PolicyVersion 数组的 Document 字段中返回的 JSON 策略文档。在该策略文档中，Arnav 在 ec2 命名空间中搜索操作。如果在该策略中没有其他命名空间中的操作，则他将该策略与受影响的身份（用户、组和角色）分离。然后，他删除该策略。在这种情况下，策略确实包含 Amazon DynamoDB 和 Amazon S3 服务。因此，Arnav 从文档中删除 Amazon EC2 操作并保存更改。然后，他使用以下命令通过文档的新版本更新策略并将该版本设置为默认策略版本。

```
aws iam create-policy-version --policy-arn arn:aws:iam::123456789012:policy/
ExamplePolicy1 --policy-document file://UpdatedPolicy.json --set-as-default
```

ExamplePolicy1 策略现已更新以删除对不必要的 Amazon EC2 服务的访问权限。

其他 IAM 应用场景

有关 IAM 资源（用户、组、角色或策略）上次尝试访问服务的时间的信息可在您完成以下任何任务时为您提供帮助：

- Policies（策略）- 编辑现有的客户托管策略或内联策略以删除权限
- Policies（策略）- 将内联策略转换为托管策略，然后将其删除
- Policies（策略）- 向现有策略添加显式拒绝
- Policies（策略）- 将托管策略与身份（用户、组或角色）分离
- Entities（实体）- 设置权限边界以控制实体（用户或角色）可以拥有的最大权限
- Groups（组）- 从组中删除用户

使用信息来优化组织部门的权限

您可以在 Amazon Organizations 中使用上次访问的信息优化组织部门 (OU) 的权限。

例如，John Stiles 是一个 Amazon Organizations 管理员。他负责确保具有公司 Amazon Web Services 账户 的人员没有多余的权限。作为定期安全审核的一部分，他检查组织的权限。他的 `Development` OU 包含经常用于测试新 Amazon 服务的账户。John 决定定期查看超过 180 天未访问的服务的报告。然后，他删除 OU 成员访问这些服务的权限。

John 使用管理账户凭证登录到 IAM 控制台。在 IAM 控制台中，他找到 Amazon Organizations OU 的 `Development` 数据。他查看 Service access report (服务访问报告) 表，并发现两个超过 180 天（首选的时段）未访问的 Amazon 服务。他记得为开发团队添加了权限以访问 Amazon Lex 和 Amazon Database Migration Service。John 联系开发团队，并确认他们不再有测试这些服务的业务需求。

John 现在已准备好处理上次访问的信息。他选择 Edit in Amazon Organizations（在 Amazon Organizations 中编辑），系统提醒他 SCP 将附加到多个实体。他选择 Continue (继续)。在 Amazon Organizations 中，他检查目标以了解 SCP 附加到哪些 Amazon Organizations 实体。所有实体均位于 `Development` OU 中。

John 决定在 `NewServiceTest` SCP 中拒绝访问 Amazon Lex 和 Amazon Database Migration Service 操作。该操作将删除这些对这些服务的不必要访问。

上次访问 IAM 操作信息的服务和操作

下表列出了将显示 上次访问 IAM 操作信息 的 Amazon 服务。有关每项服务中的操作列表，请参阅《服务授权参考》中的 Amazon 服务的操作、资源和条件键。

Amazon 以 JSON 格式提供操作的最后访问信息，以简化策略管理工作流程的自动化。借助服务参考信息，您可以从机器可读文件中跨 Amazon Web Services 服务 获取操作的最后访问信息。有关更多信息，请参阅《服务授权参考》中的用于编程访问的简化 Amazon Web Services 服务信息。

服务	服务前缀
Amazon Identity and Access Management and Access Analyzer	access-analyzer
Amazon 账户管理	账户
Amazon Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow

服务	服务前缀
Amazon Amplify	amplify
Amazon Amplify UI Builder	amplifyuibuilder
Amazon AppIntegrations	app-integrations
Amazon AppConfig	appconfig
Amazon AppFlow	appflow
Amazon Application Cost Profiler	application-cost-profiler
Amazon CloudWatch Application Insights	applicationinsights
Amazon App Mesh	appmesh
Amazon WorkSpaces Applications	appstream
Amazon AppSync	appsync
Amazon Managed Service for Prometheus	aps
Amazon Athena	athena
Amazon Audit Manager	auditmanager
Amazon Auto Scaling	自动扩缩
Amazon Web Services Marketplace	aws-marketplace
Amazon Backup	备份
Amazon Batch	批处理
Amazon Braket	Braket
Amazon Budgets	预算
Amazon Cloud9	cloud9

服务	服务前缀
Amazon CloudFormation	cloudformation
Amazon CloudFront	cloudfront
Amazon CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch
Amazon CloudTrail	cloudtrail
Amazon CloudWatch	cloudwatch
Amazon CodeArtifact	codeartifact
Amazon CodeDeploy	codedeploy
Amazon CodeGuru Profiler	codeguru-profiler
Amazon CodeGuru Reviewer	codeguru-reviewer
Amazon CodePipeline	codepipeline
AWS CodeStar	codestar
AWS CodeStar 通知	codestar-notifications
Amazon Cognito 身份	cognito-identity
Amazon Cognito 用户群体	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
Amazon Compute Optimizer	compute-optimizer
Amazon Config	config

服务	服务前缀
Amazon Connect Customer	connect
Amazon 成本和使用情况报告	cur
Amazon Glue DataBrew	databrew
Amazon Web Services Data Exchange	dataexchange
Amazon Data Pipeline	datapipeline
DynamoDB Accelerator	dax
Amazon Device Farm	devicefarm
Amazon DevOps Guru	devops-guru
Amazon Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm
Amazon Database Migration Service	dms
Amazon DocumentDB Elastic Clusters	docdb-elastic
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	ec2
Amazon Elastic Container Registry	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon ElastiCache	elasticache

服务	服务前缀
Amazon Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing
Amazon Elastic Transcoder	elastictranscoder
Amazon EMR 在 EKS 上（EMR 容器）	emr-containers
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch Service	es
Amazon EventBridge	events
Amazon CloudWatch Evidently	evidently
Amazon FinSpace	finspace
Amazon Data Firehose	Firehose
Amazon Fault Injection Service	fis
Amazon Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift Servers	GameLift
Amazon Location Service	geo
Amazon Glacier	glacier
Amazon Managed Grafana	grafana
Amazon IoT Greengrass	greengrass

服务	服务前缀
Amazon Ground Station	groundstation
Amazon GuardDuty	guardduty
Amazon HealthLake	healthlake
Amazon Honeycode	honeycode
Amazon Identity and Access Management	IAM
Amazon 身份存储	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector
Amazon Inspector	inspector2
Amazon IoT	iot
Amazon IoT Analytics	iotanalytics
Amazon IoT Core Device Advisor	iotdeviceadvisor
Amazon IoT Events	iotevents
Amazon IoT Fleet Hub	iotfleethub
Amazon IoT SiteWise	iotsitewise
Amazon IoT TwinMaker	iottwinmaker
Amazon IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka

服务	服务前缀
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis
Amazon Kinesis Analytics V2	kinesisanalytics
Amazon Key Management Service	kms
Amazon Lambda	lambda
Amazon Lex	Lex
Amazon License Manager Linux Subscriptions Manager	license-manager-linux-subscriptions
Amazon Lightsail	lightsail
Amazon CloudWatch Logs	logs
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
Amazon Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore

服务	服务前缀
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB	memorydb
Amazon Transform MGN	mgn
Amazon Migration Hub	mgh
Amazon Migration Hub 策略建议	migration hub-strategy
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
Amazon Network Manager	networkmanager
Amazon Nimble Studio	nimble
Amazon HealthOmics	omics
Amazon OpsWorks	opsworks
Amazon OpsWorks CM	opsworks-cm
Amazon Outposts	outposts
Amazon Organizations	组织
Amazon Panorama	panorama
Amazon 性能详情	pi
Amazon EventBridge Pipes	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	配置文件
Amazon QLDB	qldb

服务	服务前缀
Amazon Resource Access Manager	ram
Amazon 回收站	rbin
Amazon Relational Database Service	rds
Amazon Redshift	redshift
Amazon Redshift Data API	redshift-data
Amazon Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
Amazon Resilience Hub	resiliencehub
Amazon 资源探索器	resource-explorer-2
Amazon Resource Groups	resource-groups
Amazon RoboMaker	robomaker
Amazon Identity and Access Management Roles Anywhere	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery 控制	route53-recovery-control-config
Amazon Route 53 Recovery 就绪性	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
Amazon CloudWatch RUM	rum
Amazon Simple Storage Service	S3
Amazon S3 on Outposts	s3-outposts

服务	服务前缀
Amazon SageMaker AI 地理空间功能	sagemaker-geospatial
Savings Plans	savingsplans
Amazon EventBridge Schemas	schemas
Amazon SimpleDB	sdb
Amazon Secrets Manager	secretsmanager
Amazon Security Hub CSPM	securityhub
Amazon Security Lake	securitylake
Amazon Serverless Application Repository	serverlessrepo
Amazon Service Catalog	servicecatalog
Amazon Cloud Map	servicediscovery
服务配额	servicequotas
Amazon Simple Email Service	ses
Amazon Shield	shield
Amazon Signer	signer
Amazon SimSpace Weaver	simspaceweaver
Amazon Server Migration Service	sms
Amazon Pinpoint 短信和语音服务	sms-voice
Amazon Snowball Edge	snowball
Amazon Simple Queue Service	sqs
Amazon Systems Manager	ssm

服务	服务前缀
Amazon Systems Manager Incident Manager	ssm-incidents
适用于 SAP 的 Amazon Systems Manager	ssm-sap
Amazon Step Functions	states
Amazon Security Token Service	sts
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
Amazon Resource Groups Tagging API	标签
Amazon Textract	textract
Amazon Timestream	timestream
Amazon 电信网络生成器	tnb
Amazon Transcribe	transcribe
Amazon Transfer Family	转移
Amazon Translate	translate
Connect Customer Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
Amazon WAFV2	wafv2
Amazon Well-Architected Tool	wellarchitected
Amazon Connect Customer Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	工作区

服务	服务前缀
Amazon X-Ray	xray

上次访问操作信息的操作

下表列出了可获得上次访问操作信息的操作。

> **Important**
>
> `iam:UpdateAccountName` 操作将于 2026 年 4 月 22 日弃用。2026 年 4 月 22 日之后，只有 `account:PutAccountName` 权限才能控制账户名称更新访问权限。我们强烈建议更新任何控制账户名称更新的服务控制策略 (SCP)，以使用 `account:PutAccountName` 权限。

服务前缀	操作
access-analyzer	access-analyzer:ApplyArchiveRule
	access-analyzer:CancelPolicyGeneration
	access-analyzer:CheckAccessNotGranted
	access-analyzer:CheckNoNewAccess
	access-analyzer:CheckNoPublicAccess
	access-analyzer:CreateAccessPreview
	access-analyzer:CreateAnalyzer
	access-analyzer:CreateArchiveRule
	access-analyzer:DeleteAnalyzer
	access-analyzer:DeleteArchiveRule
	access-analyzer:GenerateFindingRecommendation
	access-analyzer:GetAccessPreview

服务前缀	操作
	access-analyzer:GetAnalyzedResource
	access-analyzer:GetAnalyzer
	access-analyzer:GetArchiveRule
	access-analyzer:GetFinding
	access-analyzer:GetFindingRecommendation
	access-analyzer:GetFindingsStatistics
	access-analyzer:GetGeneratedPolicy
	access-analyzer:ListAccessPreviewFindings
	access-analyzer:ListAccessPreviews
	access-analyzer:ListAnalyzedResources
	access-analyzer:ListAnalyzers
	access-analyzer:ListArchiveRules
	access-analyzer:ListFindings
	access-analyzer:ListPolicyGenerations
	access-analyzer:StartPolicyGeneration
	access-analyzer:StartResourceScan
	access-analyzer:UpdateAnalyzer
	access-analyzer:UpdateArchiveRule
	access-analyzer:UpdateFindings
	access-analyzer:ValidatePolicy

服务前缀	操作
账户	account:AcceptPrimaryEmailUpdate account:DeleteAlternateContact account:DisableRegion account:EnableRegion account:GetAccountInformation account:GetAlternateContact account:GetContactInformation account:GetGovCloudAccountInformation account:GetPrimaryEmail account:GetRegionOptStatus account:ListRegions account:PutAccountName account:PutAlternateContact account:PutContactInformation account:StartPrimaryEmailUpdate

服务前缀	操作
acm	acm:DeleteCertificate acm:DescribeCertificate acm:ExportCertificate acm:GetAccountConfiguration acm:GetCertificate acm:ImportCertificate acm:ListCertificates acm:PutAccountConfiguration acm:RenewCertificate acm:RequestCertificate acm:ResendValidationEmail acm:UpdateCertificateOptions
airflow	airflow:CreateCliToken airflow:CreateEnvironment airflow:CreateWebLoginToken airflow:DeleteEnvironment airflow:GetEnvironment airflow:ListEnvironments airflow:PublishMetrics airflow:UpdateEnvironment

服务前缀	操作
amplify	amplify:CreateApp
	amplify:CreateBackendEnvironment
	amplify:CreateBranch
	amplify:CreateDeployment
	amplify:CreateDomainAssociation
	amplify:CreateWebHook
	amplify:DeleteApp
	amplify:DeleteBackendEnvironment
	amplify:DeleteBranch
	amplify:DeleteDomainAssociation
	amplify:DeleteJob
	amplify:DeleteWebHook
	amplify:GenerateAccessLogs
	amplify:GetApp
	amplify:GetArtifactUrl
	amplify:GetBackendEnvironment
	amplify:GetBranch
	amplify:GetDomainAssociation
	amplify:GetJob
	amplify:GetWebHook
	amplify:ListApps

服务前缀	操作
	amplify:ListArtifacts amplify:ListBackendEnvironments amplify:ListBranches amplify:ListDomainAssociations amplify:ListJobs amplify:ListWebHooks amplify:StartDeployment amplify:StartJob amplify:StopJob amplify:UpdateApp amplify:UpdateBranch amplify:UpdateDomainAssociation amplify:UpdateWebHook

服务前缀	操作
amplifyuibuilder	amplifyuibuilder:CreateComponent amplifyuibuilder:CreateForm amplifyuibuilder:CreateTheme amplifyuibuilder:DeleteComponent amplifyuibuilder:DeleteForm amplifyuibuilder:DeleteTheme amplifyuibuilder:ExportComponents amplifyuibuilder:ExportThemes amplifyuibuilder:GetCodegenJob amplifyuibuilder:ListCodegenJobs amplifyuibuilder:ListComponents amplifyuibuilder:ListForms amplifyuibuilder:ListThemes amplifyuibuilder:ResetMetadataFlag amplifyuibuilder:StartCodegenJob amplifyuibuilder:UpdateComponent amplifyuibuilder:UpdateForm amplifyuibuilder:UpdateTheme

服务前缀	操作
app-integrations	app-integrations:CreateApplication
	app-integrations:CreateDataIntegration
	app-integrations:CreateDataIntegrationAssociation
	app-integrations:CreateEventIntegration
	app-integrations:DeleteApplication
	app-integrations:DeleteDataIntegration
	app-integrations:DeleteEventIntegration
	app-integrations:GetApplication
	app-integrations:GetDataIntegration
	app-integrations:GetEventIntegration
	app-integrations:ListApplicationAssociations
	app-integrations:ListApplications
	app-integrations:ListDataIntegrationAssociations
	app-integrations:ListDataIntegrations
	app-integrations:ListEventIntegrationAssociations
	app-integrations:ListEventIntegrations
	app-integrations:UpdateApplication
	app-integrations:UpdateDataIntegration
	app-integrations:UpdateDataIntegrationAssociation
	app-integrations:UpdateEventIntegration

服务前缀	操作
appconfig	appconfig:CreateApplication appconfig:CreateConfigurationProfile appconfig:CreateDeploymentStrategy appconfig:CreateEnvironment appconfig:CreateExtension appconfig:CreateExtensionAssociation appconfig:CreateHostedConfigurationVersion appconfig:DeleteApplication appconfig:DeleteConfigurationProfile appconfig:DeleteDeploymentStrategy appconfig:DeleteEnvironment appconfig:DeleteExtension appconfig:DeleteExtensionAssociation appconfig:DeleteHostedConfigurationVersion appconfig:GetAccountSettings appconfig:GetApplication appconfig:GetConfiguration appconfig:GetConfigurationProfile appconfig:GetDeployment appconfig:GetDeploymentStrategy appconfig:GetEnvironment

服务前缀	操作
	appconfig:GetExtension appconfig:GetExtensionAssociation appconfig:GetHostedConfigurationVersion appconfig:ListApplications appconfig:ListConfigurationProfiles appconfig:ListDeploymentStrategies appconfig:ListDeployments appconfig:ListEnvironments appconfig:ListExtensionAssociations appconfig:ListExtensions appconfig:ListHostedConfigurationVersions appconfig:StartDeployment appconfig:StopDeployment appconfig:UpdateAccountSettings appconfig:UpdateApplication appconfig:UpdateConfigurationProfile appconfig:UpdateDeploymentStrategy appconfig:UpdateEnvironment appconfig:UpdateExtension appconfig:UpdateExtensionAssociation appconfig:ValidateConfiguration

服务前缀	操作
appflow	appflow:CancelFlowExecutions
	appflow:CreateConnectorProfile
	appflow:CreateFlow
	appflow:DeleteConnectorProfile
	appflow:DeleteFlow
	appflow:DescribeConnector
	appflow:DescribeConnectorEntity
	appflow:DescribeConnectorProfiles
	appflow:DescribeConnectors
	appflow:DescribeFlow
	appflow:DescribeFlowExecutionRecords
	appflow:ListConnectorEntities
	appflow:ListConnectors
	appflow:ListFlows
	appflow:RegisterConnector
	appflow:ResetConnectorMetadataCache
	appflow:StartFlow
	appflow:StopFlow
	appflow:UnRegisterConnector
	appflow:UpdateConnectorProfile
	appflow:UpdateConnectorRegistration

服务前缀	操作
	appflow:UpdateFlow

服务前缀	操作
applicationinsights	applicationinsights:AddWorkload applicationinsights:CreateApplication applicationinsights:CreateComponent applicationinsights:CreateLogPattern applicationinsights:DeleteApplication applicationinsights:DeleteComponent applicationinsights:DeleteLogPattern applicationinsights:DescribeApplication applicationinsights:DescribeComponent applicationinsights:DescribeComponentConfiguration applicationinsights:DescribeComponentConfigurationRecommendation applicationinsights:DescribeLogPattern applicationinsights:DescribeObservation applicationinsights:DescribeProblem applicationinsights:DescribeProblemObservations applicationinsights:DescribeWorkload applicationinsights:ListApplications applicationinsights:ListComponents applicationinsights:ListConfigurationHistory applicationinsights:ListLogPatternSets

服务前缀	操作
	applicationinsights:ListLogPatterns applicationinsights:ListProblems applicationinsights:ListWorkloads applicationinsights:RemoveWorkload applicationinsights:UpdateApplication applicationinsights:UpdateComponent applicationinsights:UpdateComponentConfiguration applicationinsights:UpdateLogPattern applicationinsights:UpdateWorkload

服务前缀	操作
appmesh	appmesh:CreateGatewayRoute
	appmesh:CreateMesh
	appmesh:CreateRoute
	appmesh:CreateVirtualGateway
	appmesh:CreateVirtualNode
	appmesh:CreateVirtualRouter
	appmesh:CreateVirtualService
	appmesh:DeleteGatewayRoute
	appmesh:DeleteMesh
	appmesh:DeleteRoute
	appmesh:DeleteVirtualGateway
	appmesh:DeleteVirtualNode
	appmesh:DeleteVirtualRouter
	appmesh:DeleteVirtualService
	appmesh:DescribeGatewayRoute
	appmesh:DescribeMesh
	appmesh:DescribeRoute
	appmesh:DescribeVirtualGateway
	appmesh:DescribeVirtualNode
	appmesh:DescribeVirtualRouter
	appmesh:DescribeVirtualService

服务前缀	操作
	appmesh:ListGatewayRoutes
	appmesh:ListMeshes
	appmesh:ListRoutes
	appmesh:ListVirtualGateways
	appmesh:ListVirtualNodes
	appmesh:ListVirtualRouters
	appmesh:ListVirtualServices
	appmesh:StreamAggregatedResources
	appmesh:UpdateGatewayRoute
	appmesh:UpdateMesh
	appmesh:UpdateRoute
	appmesh:UpdateVirtualGateway
	appmesh:UpdateVirtualNode
	appmesh:UpdateVirtualRouter
	appmesh:UpdateVirtualService

服务前缀	操作
appstream	appstream:AssociateAppBlockBuilderAppBlock appstream:AssociateApplicationFleet appstream:AssociateApplicationToEntitlement appstream:AssociateFleet appstream:AssociateSoftwareToImageBuilder appstream:BatchAssociateUserStack appstream:BatchDisassociateUserStack appstream:CopyImage appstream:CreateAppBlock appstream:CreateAppBlockBuilder appstream:CreateAppBlockBuilderStreamingURL appstream:CreateApplication appstream:CreateDirectoryConfig appstream:CreateEntitlement appstream:CreateFleet appstream:CreateImageBuilder appstream:CreateImageBuilderStreamingURL appstream:CreateStack appstream:CreateStreamingURL appstream:CreateThemeForStack appstream:CreateUpdatedImage

服务前缀	操作
	appstream:CreateUsageReportSubscription appstream:CreateUser appstream:DeleteAppBlock appstream:DeleteAppBlockBuilder appstream:DeleteApplication appstream:DeleteDirectoryConfig appstream:DeleteEntitlement appstream:DeleteFleet appstream:DeleteImage appstream:DeleteImageBuilder appstream:DeleteImagePermissions appstream:DeleteStack appstream:DeleteThemeForStack appstream:DeleteUsageReportSubscription appstream:DeleteUser appstream:DescribeAppBlockBuilderAppBlockAssociations appstream:DescribeAppBlockBuilders appstream:DescribeAppBlocks appstream:DescribeAppLicenseUsage appstream:DescribeApplicationFleetAssociations appstream:DescribeApplications

服务前缀	操作
	appstream:DescribeDirectoryConfigs appstream:DescribeEntitlements appstream:DescribeFleets appstream:DescribeImageBuilders appstream:DescribeImagePermissions appstream:DescribeImages appstream:DescribeSessions appstream:DescribeStacks appstream:DescribeThemeForStack appstream:DescribeUsageReportSubscriptions appstream:DescribeUserStackAssociations appstream:DescribeUsers appstream:DisableUser appstream:DisassociateAppBlockBuilderAppBlock appstream:DisassociateApplicationFleet appstream:DisassociateApplicationFromEntitlement appstream:DisassociateFleet appstream:DisassociateSoftwareFromImageBuilder appstream:EnableUser appstream:ExpireSession appstream:GetExportImageTask

服务前缀	操作
	appstream:ListAssociatedFleets appstream:ListAssociatedStacks appstream:ListEntitledApplications appstream:ListExportImageTasks appstream:StartAppBlockBuilder appstream:StartFleet appstream:StartImageBuilder appstream:StartSoftwareDeploymentToImageBuilder appstream:StopAppBlockBuilder appstream:StopFleet appstream:StopImageBuilder appstream:UpdateAppBlockBuilder appstream:UpdateApplication appstream:UpdateDirectoryConfig appstream:UpdateEntitlement appstream:UpdateFleet appstream:UpdateImagePermissions appstream:UpdateStack appstream:UpdateThemeForStack

服务前缀	操作
appsync	appsync:AssociateApi
	appsync:AssociateMergedGraphqlApi
	appsync:AssociateSourceGraphqlApi
	appsync:CreateApi
	appsync:CreateApiCache
	appsync:CreateApiKey
	appsync:CreateChannelNamespace
	appsync:CreateDataSource
	appsync:CreateDomainName
	appsync:CreateFunction
	appsync:CreateGraphqlApi
	appsync:CreateResolver
	appsync:CreateType
	appsync:DeleteApi
	appsync:DeleteApiCache
	appsync:DeleteApiKey
	appsync:DeleteChannelNamespace
	appsync:DeleteDataSource
	appsync:DeleteDomainName
	appsync:DeleteFunction
	appsync:DeleteGraphqlApi

服务前缀	操作
	appsync:DeleteResolver
	appsync:DeleteType
	appsync:DisassociateApi
	appsync:DisassociateMergedGraphqlApi
	appsync:DisassociateSourceGraphqlApi
	appsync:EvaluateCode
	appsync:EvaluateMappingTemplate
	appsync:FlushApiCache
	appsync:GetApi
	appsync:GetApiAssociation
	appsync:GetApiCache
	appsync:GetChannelNamespace
	appsync:GetDataSource
	appsync:GetDataSourceIntrospection
	appsync:GetDomainName
	appsync:GetFunction
	appsync:GetGraphqlApi
	appsync:GetGraphqlApiEnvironmentVariables
	appsync:GetIntrospectionSchema
	appsync:GetResolver
	appsync:GetSchemaCreationStatus

服务前缀	操作
	appsync:GetSourceApiAssociation
	appsync:GetType
	appsync:ListApiKeys
	appsync:ListApis
	appsync:ListChannelNamespaces
	appsync:ListDataSources
	appsync:ListDomainNames
	appsync:ListFunctions
	appsync:ListGraphqlApis
	appsync:ListResolvers
	appsync:ListResolversByFunction
	appsync:ListSourceApiAssociations
	appsync:ListTypes
	appsync:ListTypesByAssociation
	appsync:PutGraphqlApiEnvironmentVariables
	appsync:StartDataSourceIntrospection
	appsync:StartSchemaCreation
	appsync:StartSchemaMerge
	appsync:UpdateApi
	appsync:UpdateApiCache
	appsync:UpdateApiKey

服务前缀	操作
	appsync:UpdateChannelNamespace appsync:UpdateDataSource appsync:UpdateDomainName appsync:UpdateFunction appsync:UpdateGraphqlApi appsync:UpdateResolver appsync:UpdateSourceApiAssociation appsync:UpdateType

服务前缀	操作
aps	aps:CreateAlertManagerDefinition aps:CreateAnomalyDetector aps:CreateLoggingConfiguration aps:CreateQueryLoggingConfiguration aps:CreateRuleGroupsNamespace aps:CreateWorkspace aps:DeleteAlertManagerDefinition aps:DeleteAnomalyDetector aps:DeleteLoggingConfiguration aps:DeleteQueryLoggingConfiguration aps:DeleteResourcePolicy aps:DeleteRuleGroupsNamespace aps:DeleteScraper aps:DeleteScraperLoggingConfiguration aps:DeleteWorkspace aps:DescribeAlertManagerDefinition aps:DescribeAnomalyDetector aps:DescribeLoggingConfiguration aps:DescribeQueryLoggingConfiguration aps:DescribeResourcePolicy aps:DescribeRuleGroupsNamespace

服务前缀	操作
	aps:DescribeScraper aps:DescribeScraperLoggingConfiguration aps:DescribeWorkspace aps:DescribeWorkspaceConfiguration aps:GetDefaultScraperConfiguration aps:ListAnomalyDetectors aps:ListRuleGroupsNamespaces aps:ListScrapers aps:ListWorkspaces aps:PutAlertManagerDefinition aps:PutAnomalyDetector aps:PutResourcePolicy aps:PutRuleGroupsNamespace aps:UpdateLoggingConfiguration aps:UpdateQueryLoggingConfiguration aps:UpdateScraper aps:UpdateScraperLoggingConfiguration aps:UpdateWorkspaceAlias aps:UpdateWorkspaceConfiguration

服务前缀	操作
athena	athena:BatchGetNamedQuery
	athena:BatchGetPreparedStatement
	athena:BatchGetQueryExecution
	athena:CancelCapacityReservation
	athena:CreateCapacityReservation
	athena:CreateDataCatalog
	athena:CreateNamedQuery
	athena:CreateNotebook
	athena:CreatePreparedStatement
	athena:CreatePresignedNotebookUrl
	athena:CreateWorkGroup
	athena:DeleteCapacityReservation
	athena:DeleteDataCatalog
	athena:DeleteNamedQuery
	athena:DeleteNotebook
	athena:DeletePreparedStatement
	athena:DeleteWorkGroup
	athena:ExportNotebook
	athena:GetCalculationExecution
	athena:GetCalculationExecutionCode
	athena:GetCalculationExecutionStatus

服务前缀	操作
	athena:GetCapacityAssignmentConfiguration
	athena:GetCapacityReservation
	athena:GetDataCatalog
	athena:GetDatabase
	athena:GetNamedQuery
	athena:GetNotebookMetadata
	athena:GetPreparedStatement
	athena:GetQueryExecution
	athena:GetQueryResults
	athena:GetQueryResultsStream
	athena:GetQueryRuntimeStatistics
	athena:GetResourceDashboard
	athena:GetSession
	athena:GetSessionEndpoint
	athena:GetSessionStatus
	athena:GetTableMetadata
	athena:GetWorkGroup
	athena:ImportNotebook
	athena:ListApplicationDPUSizes
	athena:ListCalculationExecutions
	athena:ListCapacityReservations

服务前缀	操作
	athena:ListDataCatalogs
	athena:ListDatabases
	athena:ListEngineVersions
	athena:ListExecutors
	athena:ListNamedQueries
	athena:ListNotebookMetadata
	athena:ListNotebookSessions
	athena:ListPreparedStatements
	athena:ListQueryExecutions
	athena:ListSessions
	athena:ListTableMetadata
	athena:ListWorkGroups
	athena:PutCapacityAssignmentConfiguration
	athena:StartCalculationExecution
	athena:StartQueryExecution
	athena:StartSession
	athena:StopCalculationExecution
	athena:StopQueryExecution
	athena:TerminateSession
	athena:UpdateCapacityReservation
	athena:UpdateDataCatalog

服务前缀	操作
	athena:UpdateNamedQuery
	athena:UpdateNotebook
	athena:UpdateNotebookMetadata
	athena:UpdatePreparedStatement
	athena:UpdateWorkGroup

服务前缀	操作
auditmanager	auditmanager:AssociateAssessmentReportEvidenceFolder auditmanager:BatchAssociateAssessmentReportEvidence auditmanager:BatchCreateDelegationByAssessment auditmanager:BatchDeleteDelegationByAssessment auditmanager:BatchDisassociateAssessmentReportEvidence auditmanager:BatchImportEvidenceToAssessmentControl auditmanager:CreateAssessment auditmanager:CreateAssessmentFramework auditmanager:CreateAssessmentReport auditmanager:CreateControl auditmanager:DeleteAssessment auditmanager:DeleteAssessmentFramework auditmanager:DeleteAssessmentFrameworkShare auditmanager:DeleteAssessmentReport auditmanager:DeleteControl auditmanager:DeregisterAccount auditmanager:DeregisterOrganizationAdminAccount auditmanager:DisassociateAssessmentReportEvidenceFolder auditmanager:GetAccountStatus auditmanager:GetAssessment auditmanager:GetAssessmentFramework

服务前缀	操作
	auditmanager:GetAssessmentReportUrl auditmanager:GetChangeLogs auditmanager:GetControl auditmanager:GetDelegations auditmanager:GetEvidence auditmanager:GetEvidenceByEvidenceFolder auditmanager:GetEvidenceFileUploadUrl auditmanager:GetEvidenceFolder auditmanager:GetEvidenceFoldersByAssessment auditmanager:GetEvidenceFoldersByAssessmentControl auditmanager:GetInsights auditmanager:GetInsightsByAssessment auditmanager:GetOrganizationAdminAccount auditmanager:GetServicesInScope auditmanager:GetSettings auditmanager:ListAssessmentControlInsightsByControlDomain auditmanager:ListAssessmentFrameworkShareRequests auditmanager:ListAssessmentFrameworks auditmanager:ListAssessmentReports auditmanager:ListAssessments auditmanager:ListControlDomainInsights

服务前缀	操作
	auditmanager:ListControlDomainInsightsByAssessment auditmanager:ListControlInsightsByControlDomain auditmanager:ListControls auditmanager:ListKeywordsForDataSource auditmanager:ListNotifications auditmanager:RegisterAccount auditmanager:RegisterOrganizationAdminAccount auditmanager:StartAssessmentFrameworkShare auditmanager:UpdateAssessment auditmanager:UpdateAssessmentControl auditmanager:UpdateAssessmentControlSetStatus auditmanager:UpdateAssessmentFramework auditmanager:UpdateAssessmentFrameworkShare auditmanager:UpdateAssessmentStatus auditmanager:UpdateControl auditmanager:UpdateSettings auditmanager:ValidateAssessmentReportIntegrity

服务前缀	操作
自动扩缩	autoscaling:AttachInstances
	autoscaling:AttachLoadBalancerTargetGroups
	autoscaling:AttachLoadBalancers
	autoscaling:AttachTrafficSources
	autoscaling:BatchDeleteScheduledAction
	autoscaling:BatchPutScheduledUpdateGroupAction
	autoscaling:CancelInstanceRefresh
	autoscaling:CompleteLifecycleAction
	autoscaling:CreateAutoScalingGroup
	autoscaling:CreateLaunchConfiguration
	autoscaling:DeleteAutoScalingGroup
	autoscaling:DeleteLaunchConfiguration
	autoscaling:DeleteLifecycleHook
	autoscaling:DeleteNotificationConfiguration
	autoscaling:DeletePolicy
	autoscaling:DeleteScheduledAction
	autoscaling:DeleteWarmPool
	autoscaling:DescribeAccountLimits
	autoscaling:DescribeAdjustmentTypes
	autoscaling:DescribeAutoScalingGroups
	autoscaling:DescribeAutoScalingInstances

服务前缀	操作
	autoscaling:DescribeAutoScalingNotificationTypes
	autoscaling:DescribeInstanceRefreshes
	autoscaling:DescribeLaunchConfigurations
	autoscaling:DescribeLifecycleHookTypes
	autoscaling:DescribeLifecycleHooks
	autoscaling:DescribeLoadBalancerTargetGroups
	autoscaling:DescribeLoadBalancers
	autoscaling:DescribeMetricCollectionTypes
	autoscaling:DescribeNotificationConfigurations
	autoscaling:DescribePolicies
	autoscaling:DescribeScalingActivities
	autoscaling:DescribeScalingProcessTypes
	autoscaling:DescribeScheduledActions
	autoscaling:DescribeTerminationPolicyTypes
	autoscaling:DescribeTrafficSources
	autoscaling:DescribeWarmPool
	autoscaling:DetachInstances
	autoscaling:DetachLoadBalancerTargetGroups
	autoscaling:DetachLoadBalancers
	autoscaling:DetachTrafficSources
	autoscaling:DisableMetricsCollection

服务前缀	操作
	autoscaling:EnableMetricsCollection autoscaling:EnterStandby autoscaling:ExecutePolicy autoscaling:ExitStandby autoscaling:GetPredictiveScalingForecast autoscaling:PutLifecycleHook autoscaling:PutNotificationConfiguration autoscaling:PutScalingPolicy autoscaling:PutScheduledUpdateGroupAction autoscaling:PutWarmPool autoscaling:RecordLifecycleActionHeartbeat autoscaling:ResumeProcesses autoscaling:RollbackInstanceRefresh autoscaling:SetDesiredCapacity autoscaling:SetInstanceHealth autoscaling:SetInstanceProtection autoscaling:StartInstanceRefresh autoscaling:SuspendProcesses autoscaling:TerminateInstanceInAutoScalingGroup autoscaling:UpdateAutoScalingGroup
aws-marketplace	aws-marketplace:GetEntitlements

服务前缀	操作
备份	backup:CancelLegalHold
	backup:CreateBackupPlan
	backup:CreateBackupSelection
	backup:CreateBackupVault
	backup:CreateFramework
	backup:CreateLegalHold
	backup:CreateLogicallyAirGappedBackupVault
	backup:CreateReportPlan
	backup:CreateRestoreAccessBackupVault
	backup:CreateRestoreTestingPlan
	backup:CreateRestoreTestingSelection
	backup:CreateTieringConfiguration
	backup:DeleteBackupPlan
	backup:DeleteBackupSelection
	backup:DeleteBackupVault
	backup:DeleteBackupVaultAccessPolicy
	backup:DeleteBackupVaultLockConfiguration
	backup:DeleteBackupVaultNotifications
	backup:DeleteFramework
	backup:DeleteRecoveryPoint
	backup:DeleteReportPlan

服务前缀	操作
	backup:DeleteRestoreTestingPlan
	backup:DeleteRestoreTestingSelection
	backup:DeleteTieringConfiguration
	backup:DescribeBackupJob
	backup:DescribeBackupVault
	backup:DescribeCopyJob
	backup:DescribeFramework
	backup:DescribeGlobalSettings
	backup:DescribeProtectedResource
	backup:DescribeRecoveryPoint
	backup:DescribeRegionSettings
	backup:DescribeReportJob
	backup:DescribeReportPlan
	backup:DescribeRestoreJob
	backup:DescribeScanJob
	backup:DisassociateRecoveryPoint
	backup:DisassociateRecoveryPointFromParent
	backup:ExportBackupPlanTemplate
	backup:GetBackupPlan
	backup:GetBackupPlanFromJSON
	backup:GetBackupPlanFromTemplate

服务前缀	操作
	backup:GetBackupSelection
	backup:GetBackupVaultAccessPolicy
	backup:GetBackupVaultNotifications
	backup:GetLegalHold
	backup:GetRecoveryPointRestoreMetadata
	backup:GetRestoreJobMetadata
	backup:GetRestoreTestingInferredMetadata
	backup:GetRestoreTestingPlan
	backup:GetRestoreTestingSelection
	backup:GetSupportedResourceTypes
	backup:GetTieringConfiguration
	backup:ListBackupJobSummaries
	backup:ListBackupJobs
	backup:ListBackupPlanTemplates
	backup:ListBackupPlanVersions
	backup:ListBackupPlans
	backup:ListBackupSelections
	backup:ListBackupVaults
	backup:ListCopyJobSummaries
	backup:ListCopyJobs
	backup:ListFrameworks

服务前缀	操作
	backup:ListIndexedRecoveryPoints
	backup:ListLegalHolds
	backup:ListProtectedResources
	backup:ListRecoveryPointsByBackupVault
	backup:ListRecoveryPointsByLegalHold
	backup:ListRecoveryPointsByResource
	backup:ListReportJobs
	backup:ListReportPlans
	backup:ListRestoreAccessBackupVaults
	backup:ListRestoreJobSummaries
	backup:ListRestoreJobs
	backup:ListRestoreJobsByProtectedResource
	backup:ListRestoreTestingPlans
	backup:ListRestoreTestingSelections
	backup:ListScanJobSummaries
	backup:ListScanJobs
	backup:ListTieringConfigurations
	backup:PutBackupVaultAccessPolicy
	backup:PutBackupVaultLockConfiguration
	backup:PutBackupVaultNotifications
	backup:PutRestoreValidationResult

服务前缀	操作
	backup:StartBackupJob
	backup:StartCopyJob
	backup:StartReportJob
	backup:StartRestoreJob
	backup:StopBackupJob
	backup:UpdateBackupPlan
	backup:UpdateFramework
	backup:UpdateGlobalSettings
	backup:UpdateRecoveryPointLifecycle
	backup:UpdateRegionSettings
	backup:UpdateReportPlan
	backup:UpdateRestoreTestingPlan
	backup:UpdateRestoreTestingSelection
	backup:UpdateTieringConfiguration

服务前缀	操作
批处理	batch:CancelJob batch:CreateComputeEnvironment batch:CreateConsumableResource batch:CreateJobQueue batch:CreateSchedulingPolicy batch:CreateServiceEnvironment batch:DeleteComputeEnvironment batch:DeleteConsumableResource batch:DeleteJobQueue batch:DeleteSchedulingPolicy batch:DeleteServiceEnvironment batch:DeregisterJobDefinition batch:DescribeComputeEnvironments batch:DescribeConsumableResource batch:DescribeJobDefinitions batch:DescribeJobQueues batch:DescribeJobs batch:DescribeSchedulingPolicies batch:DescribeServiceEnvironments batch:DescribeServiceJob batch:GetJobQueueSnapshot

服务前缀	操作
	batch:ListConsumableResources batch:ListJobs batch:ListJobsByConsumableResource batch:ListSchedulingPolicies batch:ListServiceJobs batch:RegisterJobDefinition batch:SubmitJob batch:SubmitServiceJob batch:TerminateJob batch:TerminateServiceJob batch:UpdateComputeEnvironment batch:UpdateConsumableResource batch:UpdateJobQueue batch:UpdateSchedulingPolicy batch:UpdateServiceEnvironment

服务前缀	操作
Braket	braket:CancelJob braket:CancelQuantumTask braket:CreateJob braket:CreateQuantumTask braket:CreateSpendingLimit braket:GetDevice braket:GetJob braket:GetQuantumTask braket:SearchDevices braket:SearchJobs braket:SearchQuantumTasks braket:SearchSpendingLimits

服务前缀	操作
预算	budgets:CreateBudgetAction budgets:DeleteBudgetAction budgets:DescribeBudgetAction budgets:DescribeBudgetActionHistories budgets:DescribeBudgetActionsForAccount budgets:DescribeBudgetActionsForBudget budgets:ExecuteBudgetAction budgets:ModifyBudget budgets:UpdateBudgetAction budgets:ViewBudget
cloud9	cloud9:CreateEnvironmentEC2 cloud9:CreateEnvironmentMembership cloud9:DeleteEnvironment cloud9:DeleteEnvironmentMembership cloud9:DescribeEnvironmentMemberships cloud9:DescribeEnvironmentStatus cloud9:DescribeEnvironments cloud9:ListEnvironments cloud9:UpdateEnvironment cloud9:UpdateEnvironmentMembership

服务前缀	操作
cloudformation	cloudformation:BatchDescribeTypeConfigurations cloudformation:CancelUpdateStack cloudformation:ContinueUpdateRollback cloudformation:CreateChangeSet cloudformation:CreateGeneratedTemplate cloudformation:CreateStack cloudformation:CreateStackInstances cloudformation:CreateStackSet cloudformation:DeactivateType cloudformation:DeleteChangeSet cloudformation:DeleteGeneratedTemplate cloudformation:DeleteStack cloudformation:DeleteStackInstances cloudformation:DeleteStackSet cloudformation:DeregisterType cloudformation:DescribeAccountLimits cloudformation:DescribeChangeSet cloudformation:DescribeChangeSetHooks cloudformation:DescribeEvents cloudformation:DescribeGeneratedTemplate cloudformation:DescribeOrganizationsAccess

服务前缀	操作
	cloudformation:DescribePublisher cloudformation:DescribeResourceScan cloudformation:DescribeStackDriftDetectionStatus cloudformation:DescribeStackEvents cloudformation:DescribeStackInstance cloudformation:DescribeStackResource cloudformation:DescribeStackResourceDrifts cloudformation:DescribeStackResources cloudformation:DescribeStackSet cloudformation:DescribeStackSetOperation cloudformation:DescribeStacks cloudformation:DescribeType cloudformation:DescribeTypeRegistration cloudformation:DetectStackDrift cloudformation:DetectStackResourceDrift cloudformation:DetectStackSetDrift cloudformation:EstimateTemplateCost cloudformation:ExecuteChangeSet cloudformation:GetGeneratedTemplate cloudformation:GetHookResult cloudformation:GetStackPolicy

服务前缀	操作
	cloudformation:GetTemplate cloudformation:GetTemplateSummary cloudformation:ImportStacksToStackSet cloudformation:ListChangeSets cloudformation:ListExports cloudformation:ListGeneratedTemplates cloudformation:ListHookResults cloudformation:ListImports cloudformation:ListResourceScanRelatedResources cloudformation:ListResourceScanResources cloudformation:ListResourceScans cloudformation:ListStackInstanceResourceDrifts cloudformation:ListStackInstances cloudformation:ListStackRefactors cloudformation:ListStackResources cloudformation:ListStackSetAutoDeploymentTargets cloudformation:ListStackSetOperationResults cloudformation:ListStackSetOperations cloudformation:ListStackSets cloudformation:ListTypeRegistrations cloudformation:ListTypeVersions

服务前缀	操作
	cloudformation:ListTypes cloudformation:PublishType cloudformation:RecordHandlerProgress cloudformation:RegisterPublisher cloudformation:RegisterType cloudformation:RollbackStack cloudformation:SetStackPolicy cloudformation:SetTypeConfiguration cloudformation:SetTypeDefaultVersion cloudformation:SignalResource cloudformation:StartResourceScan cloudformation:StopStackSetOperation cloudformation:TestType cloudformation:UpdateGeneratedTemplate cloudformation:UpdateStack cloudformation:UpdateStackInstances cloudformation:UpdateStackSet cloudformation:UpdateTerminationProtection cloudformation:ValidateTemplate

服务前缀	操作
cloudfront	cloudfront:AssociateAlias
	cloudfront:AssociateDistributionTenantWebACL
	cloudfront:AssociateDistributionWebACL
	cloudfront:CreateCachePolicy
	cloudfront:CreateCloudFrontOriginAccessIdentity
	cloudfront:CreateConnectionFunction
	cloudfront:CreateContinuousDeploymentPolicy
	cloudfront:CreateDistributionTenant
	cloudfront:CreateFieldLevelEncryptionConfig
	cloudfront:CreateFieldLevelEncryptionProfile
	cloudfront:CreateFunction
	cloudfront:CreateInvalidation
	cloudfront:CreateKeyGroup
	cloudfront:CreateKeyValueStore
	cloudfront:CreateMonitoringSubscription
	cloudfront:CreateOriginAccessControl
	cloudfront:CreateOriginRequestPolicy
	cloudfront:CreatePublicKey
	cloudfront:CreateRealtimeLogConfig
	cloudfront:CreateResponseHeadersPolicy
	cloudfront:CreateTrustStore

服务前缀	操作
	cloudfront:DeleteAnycastIpList
	cloudfront:DeleteCachePolicy
	cloudfront:DeleteCloudFrontOriginAccessIdentity
	cloudfront:DeleteConnectionFunction
	cloudfront:DeleteConnectionGroup
	cloudfront:DeleteContinuousDeploymentPolicy
	cloudfront:DeleteDistribution
	cloudfront:DeleteDistributionTenant
	cloudfront:DeleteFieldLevelEncryptionConfig
	cloudfront:DeleteFieldLevelEncryptionProfile
	cloudfront:DeleteFunction
	cloudfront:DeleteKeyGroup
	cloudfront:DeleteKeyValueStore
	cloudfront:DeleteMonitoringSubscription
	cloudfront:DeleteOriginAccessControl
	cloudfront:DeleteOriginRequestPolicy
	cloudfront:DeletePublicKey
	cloudfront:DeleteRealtimeLogConfig
	cloudfront:DeleteResponseHeadersPolicy
	cloudfront:DeleteStreamingDistribution
	cloudfront:DeleteTrustStore

服务前缀	操作
	cloudfront:DeleteVpcOrigin
	cloudfront:DescribeFunction
	cloudfront:DescribeKeyValueStore
	cloudfront:DisassociateDistributionTenantWebACL
	cloudfront:DisassociateDistributionWebACL
	cloudfront:GetAnycastIpList
	cloudfront:GetCachePolicy
	cloudfront:GetCachePolicyConfig
	cloudfront:GetCloudFrontOriginAccessIdentity
	cloudfront:GetCloudFrontOriginAccessIdentityConfig
	cloudfront:GetContinuousDeploymentPolicy
	cloudfront:GetContinuousDeploymentPolicyConfig
	cloudfront:GetDistributionConfig
	cloudfront:GetFieldLevelEncryption
	cloudfront:GetFieldLevelEncryptionConfig
	cloudfront:GetFieldLevelEncryptionProfile
	cloudfront:GetFieldLevelEncryptionProfileConfig
	cloudfront:GetFunction
	cloudfront:GetInvalidation
	cloudfront:GetInvalidationForDistributionTenant
	cloudfront:GetKeyGroup

服务前缀	操作
	cloudfront:GetKeyGroupConfig
	cloudfront:GetMonitoringSubscription
	cloudfront:GetOriginAccessControl
	cloudfront:GetOriginAccessControlConfig
	cloudfront:GetOriginRequestPolicy
	cloudfront:GetOriginRequestPolicyConfig
	cloudfront:GetPublicKey
	cloudfront:GetPublicKeyConfig
	cloudfront:GetRealtimeLogConfig
	cloudfront:GetResponseHeadersPolicy
	cloudfront:GetResponseHeadersPolicyConfig
	cloudfront:GetStreamingDistribution
	cloudfront:GetStreamingDistributionConfig
	cloudfront:GetVpcOrigin
	cloudfront:ListAnycastIpLists
	cloudfront:ListCachePolicies
	cloudfront:ListCloudFrontOriginAccessIdentities
	cloudfront:ListConflictingAliases
	cloudfront:ListConnectionFunctions
	cloudfront:ListConnectionGroups
	cloudfront:ListContinuousDeploymentPolicies

服务前缀	操作
	cloudfront:ListDistributionTenants
	cloudfront:ListDistributionTenantsByCustomization
	cloudfront:ListDistributions
	cloudfront:ListDistributionsByAnycastIpListId
	cloudfront:ListDistributionsByCachePolicyId
	cloudfront:ListDistributionsByConnectionMode
	cloudfront:ListDistributionsByKeyGroup
	cloudfront:ListDistributionsByOriginRequestPolicyId
	cloudfront:ListDistributionsByRealtimeLogConfig
	cloudfront:ListDistributionsByResponseHeadersPolicyId
	cloudfront:ListDistributionsByVpcOriginId
	cloudfront:ListDistributionsByWebACLId
	cloudfront:ListFieldLevelEncryptionConfigs
	cloudfront:ListFieldLevelEncryptionProfiles
	cloudfront:ListFunctions
	cloudfront:ListInvalidations
	cloudfront:ListInvalidationsForDistributionTenant
	cloudfront:ListKeyGroups
	cloudfront:ListKeyValueStores
	cloudfront:ListOriginAccessControls
	cloudfront:ListOriginRequestPolicies

服务前缀	操作
	cloudfront:ListPublicKeys
	cloudfront:ListRealtimeLogConfigs
	cloudfront:ListResponseHeadersPolicies
	cloudfront:ListStreamingDistributions
	cloudfront:ListTrustStores
	cloudfront:PublishConnectionFunction
	cloudfront:PublishFunction
	cloudfront:TestConnectionFunction
	cloudfront:TestFunction
	cloudfront:UpdateAnycastIpList
	cloudfront:UpdateCachePolicy
	cloudfront:UpdateCloudFrontOriginAccessIdentity
	cloudfront:UpdateConnectionFunction
	cloudfront:UpdateConnectionGroup
	cloudfront:UpdateContinuousDeploymentPolicy
	cloudfront:UpdateDistribution
	cloudfront:UpdateDistributionTenant
	cloudfront:UpdateFieldLevelEncryptionConfig
	cloudfront:UpdateFieldLevelEncryptionProfile
	cloudfront:UpdateFunction
	cloudfront:UpdateKeyGroup

服务前缀	操作
	cloudfront:UpdateKeyValueStore cloudfront:UpdateOriginAccessControl cloudfront:UpdateOriginRequestPolicy cloudfront:UpdatePublicKey cloudfront:UpdateRealtimeLogConfig cloudfront:UpdateResponseHeadersPolicy cloudfront:UpdateTrustStore
cloudhsm	cloudhsm:CreateHsm cloudhsm:DeleteBackup cloudhsm:DeleteHsm cloudhsm:DeleteResourcePolicy cloudhsm:DescribeBackups cloudhsm:DescribeClusters cloudhsm:GetResourcePolicy cloudhsm:InitializeCluster cloudhsm:ModifyBackupAttributes cloudhsm:ModifyCluster cloudhsm:PutResourcePolicy cloudhsm:RestoreBackup

服务前缀	操作
cloudsearch	cloudsearch:BuildSuggesters
	cloudsearch:CreateDomain
	cloudsearch:DefineAnalysisScheme
	cloudsearch:DefineExpression
	cloudsearch:DefineIndexField
	cloudsearch:DefineSuggester
	cloudsearch:DeleteAnalysisScheme
	cloudsearch:DeleteDomain
	cloudsearch:DeleteExpression
	cloudsearch:DeleteIndexField
	cloudsearch:DeleteSuggester
	cloudsearch:DescribeAnalysisSchemes
	cloudsearch:DescribeAvailabilityOptions
	cloudsearch:DescribeDomainEndpointOptions
	cloudsearch:DescribeDomains
	cloudsearch:DescribeExpressions
	cloudsearch:DescribeIndexFields
	cloudsearch:DescribeScalingParameters
	cloudsearch:DescribeServiceAccessPolicies
	cloudsearch:DescribeSuggesters
	cloudsearch:IndexDocuments

服务前缀	操作
	cloudsearch:ListDomainNames
	cloudsearch:UpdateAvailabilityOptions
	cloudsearch:UpdateDomainEndpointOptions
	cloudsearch:UpdateScalingParameters
	cloudsearch:UpdateServiceAccessPolicies

服务前缀	操作
cloudtrail	cloudtrail:CancelQuery cloudtrail:CreateChannel cloudtrail:CreateDashboard cloudtrail:CreateEventDataStore cloudtrail:CreateTrail cloudtrail:DeleteChannel cloudtrail:DeleteDashboard cloudtrail:DeleteEventDataStore cloudtrail:DeleteResourcePolicy cloudtrail:DeleteTrail cloudtrail:DeregisterOrganizationDelegatedAdmin cloudtrail:DescribeQuery cloudtrail:DescribeTrails cloudtrail:DisableFederation cloudtrail:GenerateQuery cloudtrail:GetChannel cloudtrail:GetDashboard cloudtrail:GetEventConfiguration cloudtrail:GetEventDataStore cloudtrail:GetEventDataStoreData cloudtrail:GetEventSelectors

服务前缀	操作
	cloudtrail:GetImport cloudtrail:GetInsightSelectors cloudtrail:GetResourcePolicy cloudtrail:GetTrail cloudtrail:GetTrailStatus cloudtrail:ListChannels cloudtrail:ListDashboards cloudtrail:ListEventDataStores cloudtrail:ListImportFailures cloudtrail:ListImports cloudtrail:ListInsightsData cloudtrail:ListPublicKeys cloudtrail:ListQueries cloudtrail:ListTrails cloudtrail:LookupEvents cloudtrail:PutEventConfiguration cloudtrail:PutEventSelectors cloudtrail:PutInsightSelectors cloudtrail:PutResourcePolicy cloudtrail:RegisterOrganizationDelegatedAdmin cloudtrail:RestoreEventDataStore

服务前缀	操作
	cloudtrail:SearchSampleQueries cloudtrail:StartEventDataStoreIngestion cloudtrail:StartImport cloudtrail:StartLogging cloudtrail:StartQuery cloudtrail:StopEventDataStoreIngestion cloudtrail:StopImport cloudtrail:StopLogging cloudtrail:UpdateChannel cloudtrail:UpdateDashboard cloudtrail:UpdateEventDataStore cloudtrail:UpdateTrail

服务前缀	操作
cloudwatch	cloudwatch:DeleteAlarms
	cloudwatch:DeleteAnomalyDetector
	cloudwatch:DeleteDashboards
	cloudwatch:DeleteInsightRules
	cloudwatch:DeleteMetricStream
	cloudwatch:DescribeAlarmHistory
	cloudwatch:DescribeAlarms
	cloudwatch:DescribeAlarmsForMetric
	cloudwatch:DescribeAnomalyDetectors
	cloudwatch:DescribeInsightRules
	cloudwatch:DisableAlarmActions
	cloudwatch:DisableInsightRules
	cloudwatch:EnableAlarmActions
	cloudwatch:EnableInsightRules
	cloudwatch:GetDashboard
	cloudwatch:GetInsightRuleReport
	cloudwatch:GetMetricStatistics
	cloudwatch:GetMetricStream
	cloudwatch:ListDashboards
	cloudwatch:ListManagedInsightRules
	cloudwatch:ListMetricStreams

服务前缀	操作
	cloudwatch:PutAnomalyDetector
	cloudwatch:PutCompositeAlarm
	cloudwatch:PutDashboard
	cloudwatch:PutInsightRule
	cloudwatch:PutManagedInsightRules
	cloudwatch:PutMetricAlarm
	cloudwatch:PutMetricStream
	cloudwatch:SetAlarmState
	cloudwatch:StartMetricStreams
	cloudwatch:StopMetricStreams

服务前缀	操作
codeartifact	codeartifact:AssociateExternalConnection codeartifact:CopyPackageVersions codeartifact:CreateDomain codeartifact:CreateRepository codeartifact:DeleteDomain codeartifact:DeleteDomainPermissionsPolicy codeartifact:DeletePackage codeartifact:DeletePackageVersions codeartifact:DeleteRepository codeartifact:DeleteRepositoryPermissionsPolicy codeartifact:DescribeDomain codeartifact:DescribePackage codeartifact:DescribePackageVersion codeartifact:DescribeRepository codeartifact:DisassociateExternalConnection codeartifact:DisposePackageVersions codeartifact:GetAssociatedPackageGroup codeartifact:GetAuthorizationToken codeartifact:GetDomainPermissionsPolicy codeartifact:GetPackageVersionAsset codeartifact:GetPackageVersionReadme

服务前缀	操作
	codeartifact:GetRepositoryEndpoint codeartifact:GetRepositoryPermissionsPolicy codeartifact:ListDomains codeartifact:ListPackageGroups codeartifact:ListPackageVersionAssets codeartifact:ListPackageVersionDependencies codeartifact:ListPackageVersions codeartifact:ListPackages codeartifact:ListRepositories codeartifact:ListRepositoriesInDomain codeartifact:PublishPackageVersion codeartifact:PutDomainPermissionsPolicy codeartifact:PutPackageMetadata codeartifact:PutPackageOriginConfiguration codeartifact:PutRepositoryPermissionsPolicy codeartifact:ReadFromRepository codeartifact:UpdatePackageVersionsStatus codeartifact:UpdateRepository

服务前缀	操作
codedeploy	codedeploy:BatchGetApplicationRevisions
	codedeploy:BatchGetApplications
	codedeploy:BatchGetDeploymentGroups
	codedeploy:BatchGetDeploymentInstances
	codedeploy:BatchGetDeploymentTargets
	codedeploy:BatchGetDeployments
	codedeploy:BatchGetOnPremisesInstances
	codedeploy:ContinueDeployment
	codedeploy:CreateApplication
	codedeploy:CreateDeployment
	codedeploy:CreateDeploymentConfig
	codedeploy:CreateDeploymentGroup
	codedeploy:DeleteApplication
	codedeploy:DeleteDeploymentConfig
	codedeploy:DeleteDeploymentGroup
	codedeploy:DeleteGitHubAccountToken
	codedeploy:DeleteResourcesByExternalId
	codedeploy:DeregisterOnPremisesInstance
	codedeploy:GetApplication
	codedeploy:GetApplicationRevision
	codedeploy:GetDeployment

服务前缀	操作
	codedeploy:GetDeploymentConfig
	codedeploy:GetDeploymentGroup
	codedeploy:GetDeploymentInstance
	codedeploy:GetDeploymentTarget
	codedeploy:GetOnPremisesInstance
	codedeploy:ListApplicationRevisions
	codedeploy:ListApplications
	codedeploy:ListDeploymentConfigs
	codedeploy:ListDeploymentGroups
	codedeploy:ListDeploymentInstances
	codedeploy:ListDeploymentTargets
	codedeploy:ListDeployments
	codedeploy:ListGitHubAccountTokenNames
	codedeploy:ListOnPremisesInstances
	codedeploy:PutLifecycleEventHookExecutionStatus
	codedeploy:RegisterApplicationRevision
	codedeploy:RegisterOnPremisesInstance
	codedeploy:SkipWaitTimeForInstanceTermination
	codedeploy:StopDeployment
	codedeploy:UpdateApplication
	codedeploy:UpdateDeploymentGroup

服务前缀	操作
codeguru-profiler	codeguru-profiler:AddNotificationChannels codeguru-profiler:BatchGetFrameMetricData codeguru-profiler:CreateProfilingGroup codeguru-profiler:DeleteProfilingGroup codeguru-profiler:DescribeProfilingGroup codeguru-profiler:GetFindingsReportAccountSummary codeguru-profiler:GetNotificationConfiguration codeguru-profiler:GetPolicy codeguru-profiler:GetProfile codeguru-profiler:GetRecommendations codeguru-profiler:ListFindingsReports codeguru-profiler:ListProfileTimes codeguru-profiler:ListProfilingGroups codeguru-profiler:PutPermission codeguru-profiler:RemoveNotificationChannel codeguru-profiler:RemovePermission codeguru-profiler:SubmitFeedback codeguru-profiler:UpdateProfilingGroup

服务前缀	操作
codeguru-reviewer	codeguru-reviewer:AssociateRepository codeguru-reviewer:CreateCodeReview codeguru-reviewer:DescribeCodeReview codeguru-reviewer:DescribeRecommendationFeedback codeguru-reviewer:DescribeRepositoryAssociation codeguru-reviewer:DisassociateRepository codeguru-reviewer:ListCodeReviews codeguru-reviewer:ListRecommendationFeedback codeguru-reviewer:ListRecommendations codeguru-reviewer:ListRepositoryAssociations codeguru-reviewer:PutRecommendationFeedback

服务前缀	操作
codepipeline	codepipeline:AcknowledgeJob codepipeline:AcknowledgeThirdPartyJob codepipeline:CreateCustomActionType codepipeline:CreatePipeline codepipeline:DeleteCustomActionType codepipeline:DeletePipeline codepipeline:DeleteWebhook codepipeline:DeregisterWebhookWithThirdParty codepipeline:GetActionType codepipeline:GetJobDetails codepipeline:GetPipeline codepipeline:GetPipelineExecution codepipeline:GetPipelineState codepipeline:GetThirdPartyJobDetails codepipeline:ListActionExecutions codepipeline:ListActionTypes codepipeline:ListPipelineExecutions codepipeline:ListPipelines codepipeline:ListRuleExecutions codepipeline:ListRuleTypes codepipeline:ListWebhooks

服务前缀	操作
	codepipeline:OverrideStageCondition codepipeline:PollForJobs codepipeline:PollForThirdPartyJobs codepipeline:PutActionRevision codepipeline:PutApprovalResult codepipeline:PutJobFailureResult codepipeline:PutJobSuccessResult codepipeline:PutThirdPartyJobFailureResult codepipeline:PutThirdPartyJobSuccessResult codepipeline:PutWebhook codepipeline:RegisterWebhookWithThirdParty codepipeline:RollbackStage codepipeline:StartPipelineExecution codepipeline:StopPipelineExecution codepipeline:UpdateActionType codepipeline:UpdatePipeline

服务前缀	操作
codestar	codestar:AssociateTeamMember
	codestar:CreateProject
	codestar:CreateUserProfile
	codestar:DeleteProject
	codestar:DeleteUserProfile
	codestar:DescribeProject
	codestar:DescribeUserProfile
	codestar:DisassociateTeamMember
	codestar:ListProjects
	codestar:ListResources
	codestar:ListTeamMembers
	codestar:ListUserProfiles
	codestar:UpdateProject
	codestar:UpdateTeamMember
	codestar:UpdateUserProfile

服务前缀	操作
codestar-notifications	codestar-notifications:CreateNotificationRule codestar-notifications:DeleteNotificationRule codestar-notifications:DeleteTarget codestar-notifications:DescribeNotificationRule codestar-notifications:ListEventTypes codestar-notifications:ListNotificationRules codestar-notifications:ListTargets codestar-notifications:Subscribe codestar-notifications:Unsubscribe codestar-notifications:UpdateNotificationRule

服务前缀	操作
cognito-identity	cognito-identity:CreateIdentityPool
	cognito-identity:DeleteIdentities
	cognito-identity:DeleteIdentityPool
	cognito-identity:DescribeIdentity
	cognito-identity:DescribeIdentityPool
	cognito-identity:GetIdentityPoolRoles
	cognito-identity:ListIdentities
	cognito-identity:ListIdentityPools
	cognito-identity:LookupDeveloperIdentity
	cognito-identity:MergeDeveloperIdentities
	cognito-identity:SetIdentityPoolRoles
	cognito-identity:UnlinkDeveloperIdentity
	cognito-identity:UpdateIdentityPool

服务前缀	操作
cognito-idp	cognito-idp:AddCustomAttributes cognito-idp:AdminAddUserToGroup cognito-idp:AdminConfirmSignUp cognito-idp:AdminCreateUser cognito-idp:AdminDeleteUser cognito-idp:AdminDeleteUserAttributes cognito-idp:AdminDisableProviderForUser cognito-idp:AdminDisableUser cognito-idp:AdminEnableUser cognito-idp:AdminForgetDevice cognito-idp:AdminGetDevice cognito-idp:AdminGetUser cognito-idp:AdminInitiateAuth cognito-idp:AdminLinkProviderForUser cognito-idp:AdminListDevices cognito-idp:AdminListGroupsForUser cognito-idp:AdminListUserAuthEvents cognito-idp:AdminRemoveUserFromGroup cognito-idp:AdminResetUserPassword cognito-idp:AdminRespondToAuthChallenge cognito-idp:AdminSetUserMFAPreference

服务前缀	操作
	cognito-idp:AdminSetUserPassword cognito-idp:AdminSetUserSettings cognito-idp:AdminUpdateAuthEventFeedback cognito-idp:AdminUpdateDeviceStatus cognito-idp:AdminUpdateUserAttributes cognito-idp:AdminUserGlobalSignOut cognito-idp:AssociateSoftwareToken cognito-idp:ChangePassword cognito-idp:ConfirmDevice cognito-idp:ConfirmForgotPassword cognito-idp:ConfirmSignUp cognito-idp:CreateGroup cognito-idp:CreateIdentityProvider cognito-idp:CreateManagedLoginBranding cognito-idp:CreateResourceServer cognito-idp:CreateTerms cognito-idp:CreateUserImportJob cognito-idp:CreateUserPool cognito-idp:CreateUserPoolClient cognito-idp:CreateUserPoolDomain cognito-idp:DeleteGroup

服务前缀	操作
	cognito-idp:DeleteIdentityProvider cognito-idp:DeleteManagedLoginBranding cognito-idp:DeleteResourceServer cognito-idp:DeleteTerms cognito-idp:DeleteUser cognito-idp:DeleteUserAttributes cognito-idp:DeleteUserPool cognito-idp:DeleteUserPoolClient cognito-idp:DeleteUserPoolDomain cognito-idp:DescribeIdentityProvider cognito-idp:DescribeManagedLoginBranding cognito-idp:DescribeManagedLoginBrandingByClient cognito-idp:DescribeResourceServer cognito-idp:DescribeRiskConfiguration cognito-idp:DescribeTerms cognito-idp:DescribeUserImportJob cognito-idp:DescribeUserPool cognito-idp:DescribeUserPoolClient cognito-idp:DescribeUserPoolDomain cognito-idp:ForgetDevice cognito-idp:ForgotPassword

服务前缀	操作
	cognito-idp:GetCSVHeader cognito-idp:GetDevice cognito-idp:GetGroup cognito-idp:GetIdentityProviderByIdentifier cognito-idp:GetLogDeliveryConfiguration cognito-idp:GetSigningCertificate cognito-idp:GetUICustomization cognito-idp:GetUser cognito-idp:GetUserAttributeVerificationCode cognito-idp:GetUserPoolMfaConfig cognito-idp:GlobalSignOut cognito-idp:InitiateAuth cognito-idp:ListDevices cognito-idp:ListGroups cognito-idp:ListIdentityProviders cognito-idp:ListResourceServers cognito-idp:ListTerms cognito-idp:ListUserImportJobs cognito-idp:ListUserPoolClients cognito-idp:ListUserPools cognito-idp:ListUsers

服务前缀	操作
	cognito-idp:ListUsersInGroup cognito-idp:ResendConfirmationCode cognito-idp:RespondToAuthChallenge cognito-idp:RevokeToken cognito-idp:SetLogDeliveryConfiguration cognito-idp:SetRiskConfiguration cognito-idp:SetUICustomization cognito-idp:SetUserMFAPreference cognito-idp:SetUserPoolMfaConfig cognito-idp:SetUserSettings cognito-idp:SignUp cognito-idp:StartUserImportJob cognito-idp:StopUserImportJob cognito-idp:UpdateAuthEventFeedback cognito-idp:UpdateDeviceStatus cognito-idp:UpdateGroup cognito-idp:UpdateIdentityProvider cognito-idp:UpdateResourceServer cognito-idp:UpdateTerms cognito-idp:UpdateUserAttributes cognito-idp:UpdateUserPool

服务前缀	操作
	cognito-idp:UpdateUserPoolClient cognito-idp:UpdateUserPoolDomain cognito-idp:VerifySoftwareToken cognito-idp:VerifyUserAttribute
cognito-sync	cognito-sync:BulkPublish cognito-sync:DeleteDataset cognito-sync:DescribeDataset cognito-sync:DescribeIdentityPoolUsage cognito-sync:DescribeIdentityUsage cognito-sync:GetBulkPublishDetails cognito-sync:GetCognitoEvents cognito-sync:GetIdentityPoolConfiguration cognito-sync:ListDatasets cognito-sync:ListIdentityPoolUsage cognito-sync:ListRecords cognito-sync:RegisterDevice cognito-sync:SetCognitoEvents cognito-sync:SetIdentityPoolConfiguration cognito-sync:SubscribeToDataset cognito-sync:UnsubscribeFromDataset cognito-sync:UpdateRecords

服务前缀	操作
comprehendmedical	comprehendmedical:DescribeEntitiesDetectionV2Job comprehendmedical:DescribeICD10CMInferenceJob comprehendmedical:DescribePHIDetectionJob comprehendmedical:DescribeRxNormInferenceJob comprehendmedical:DescribeSNOMEDCTInferenceJob comprehendmedical:DetectEntitiesV2 comprehendmedical:DetectPHI comprehendmedical:InferICD10CM comprehendmedical:InferRxNorm comprehendmedical:InferSNOMEDCT comprehendmedical:ListEntitiesDetectionV2Jobs comprehendmedical:ListICD10CMInferenceJobs comprehendmedical:ListPHIDetectionJobs comprehendmedical:ListRxNormInferenceJobs comprehendmedical:ListSNOMEDCTInferenceJobs comprehendmedical:StartEntitiesDetectionV2Job comprehendmedical:StartICD10CMInferenceJob comprehendmedical:StartPHIDetectionJob comprehendmedical:StartRxNormInferenceJob comprehendmedical:StartSNOMEDCTInferenceJob comprehendmedical:StopEntitiesDetectionV2Job

服务前缀	操作
	comprehendmedical:StopICD10CMInferenceJob comprehendmedical:StopPHIDetectionJob comprehendmedical:StopRxNormInferenceJob comprehendmedical:StopSNOMEDCTInferenceJob

服务前缀	操作
compute-optimizer	compute-optimizer:DeleteRecommendationPreferences
	compute-optimizer:DescribeRecommendationExportJobs
	compute-optimizer:ExportAutoScalingGroupRecommendations
	compute-optimizer:ExportEBSVolumeRecommendations
	compute-optimizer:ExportEC2InstanceRecommendations
	compute-optimizer:ExportECSServiceRecommendations
	compute-optimizer:ExportIdleRecommendations
	compute-optimizer:ExportLambdaFunctionRecommendations
	compute-optimizer:ExportLicenseRecommendations
	compute-optimizer:ExportRDSDatabaseRecommendations
	compute-optimizer:GetEC2RecommendationProjectedMetrics
	compute-optimizer:GetECSServiceRecommendationProjectedMetrics
	compute-optimizer:GetEffectiveRecommendationPreferences
	compute-optimizer:GetEnrollmentStatus
	compute-optimizer:GetEnrollmentStatusesForOrganization
	compute-optimizer:GetRDSDatabaseRecommendationProjectedMetrics
	compute-optimizer:GetRecommendationPreferences
	compute-optimizer:GetRecommendationSummaries
	compute-optimizer:PutRecommendationPreferences

服务前缀	操作
	compute-optimizer:UpdateEnrollmentStatus

服务前缀	操作
config	config:BatchGetResourceConfig config:DeleteAggregationAuthorization config:DeleteConfigRule config:DeleteConfigurationAggregator config:DeleteConfigurationRecorder config:DeleteConformancePack config:DeleteDeliveryChannel config:DeleteEvaluationResults config:DeleteOrganizationConfigRule config:DeleteOrganizationConformancePack config:DeletePendingAggregationRequest config:DeleteRemediationConfiguration config:DeleteRemediationExceptions config:DeleteResourceConfig config:DeleteRetentionConfiguration config:DeleteStoredQuery config:DeliverConfigSnapshot config:DescribeAggregateComplianceByConfigRules config:DescribeAggregateComplianceByConformancePacks config:DescribeAggregationAuthorizations config:DescribeComplianceByConfigRule

服务前缀	操作
	config:DescribeComplianceByResource config:DescribeConfigRuleEvaluationStatus config:DescribeConfigRules config:DescribeConfigurationAggregatorSourcesStatus config:DescribeConfigurationAggregators config:DescribeConfigurationRecorderStatus config:DescribeConfigurationRecorders config:DescribeConformancePackCompliance config:DescribeConformancePackStatus config:DescribeConformancePacks config:DescribeDeliveryChannelStatus config:DescribeDeliveryChannels config:DescribeOrganizationConfigRuleStatuses config:DescribeOrganizationConfigRules config:DescribeOrganizationConformancePackStatuses config:DescribeOrganizationConformancePacks config:DescribePendingAggregationRequests config:DescribeRemediationConfigurations config:DescribeRemediationExceptions config:DescribeRemediationExecutionStatus config:DescribeRetentionConfigurations

服务前缀	操作
	config:GetComplianceDetailsByConfigRule config:GetComplianceDetailsByResource config:GetComplianceSummaryByConfigRule config:GetComplianceSummaryByResourceType config:GetConformancePackComplianceDetails config:GetConformancePackComplianceSummary config:GetCustomRulePolicy config:GetDiscoveredResourceCounts config:GetOrganizationConfigRuleDetailedStatus config:GetOrganizationConformancePackDetailedStatus config:GetOrganizationCustomRulePolicy config:GetResourceConfigHistory config:GetResourceEvaluationSummary config:GetStoredQuery config:ListConfigurationRecorders config:ListConformancePackComplianceScores config:ListDiscoveredResources config:ListResourceEvaluations config:ListStoredQueries config:PutConfigRule config:PutConfigurationAggregator

服务前缀	操作
	config:PutConfigurationRecorder config:PutConformancePack config:PutDeliveryChannel config:PutEvaluations config:PutExternalEvaluation config:PutOrganizationConfigRule config:PutOrganizationConformancePack config:PutRemediationConfigurations config:PutRemediationExceptions config:PutResourceConfig config:PutRetentionConfiguration config:PutStoredQuery config:SelectResourceConfig config:StartConfigRulesEvaluation config:StartConfigurationRecorder config:StartRemediationExecution config:StartResourceEvaluation config:StopConfigurationRecorder

服务前缀	操作
connect	connect:ActivateEvaluationForm
	connect:AssociateAnalyticsDataSet
	connect:AssociateApprovedOrigin
	connect:AssociateBot
	connect:AssociateContactWithUser
	connect:AssociateDefaultVocabulary
	connect:AssociateEmailAddressAlias
	connect:AssociateFlow
	connect:AssociateInstanceStorageConfig
	connect:AssociateLambdaFunction
	connect:AssociateLexBot
	connect:AssociatePhoneNumberContactFlow
	connect:AssociateQueueQuickConnects
	connect:AssociateRoutingProfileQueues
	connect:AssociateSecurityKey
	connect:AssociateUserProficiencies
	connect:BatchAssociateAnalyticsDataSet
	connect:BatchCreateDataTableValue
	connect:BatchDeleteDataTableValue
	connect:BatchDescribeDataTableValue
	connect:BatchDisassociateAnalyticsDataSet

服务前缀	操作
	connect:BatchGetFlowAssociation
	connect:BatchPutContact
	connect:BatchUpdateDataTableValue
	connect:ClaimPhoneNumber
	connect:CreateAgentStatus
	connect:CreateContact
	connect:CreateContactFlow
	connect:CreateContactFlowModule
	connect:CreateContactFlowModuleAlias
	connect:CreateContactFlowModuleVersion
	connect:CreateContactFlowVersion
	connect:CreateDataTable
	connect:CreateDataTableAttribute
	connect:CreateEmailAddress
	connect:CreateEvaluationForm
	connect:CreateHoursOfOperation
	connect:CreateInstance
	connect:CreateIntegrationAssociation
	connect:CreateParticipant
	connect:CreatePersistentContactAssociation
	connect:CreatePredefinedAttribute

服务前缀	操作
	connect:CreatePrompt
	connect:CreatePushNotificationRegistration
	connect:CreateQueue
	connect:CreateQuickConnect
	connect:CreateRoutingProfile
	connect:CreateRule
	connect:CreateSecurityProfile
	connect:CreateTaskTemplate
	connect:CreateTrafficDistributionGroup
	connect:CreateUseCase
	connect:CreateUser
	connect:CreateUserHierarchyGroup
	connect:CreateView
	connect:CreateViewVersion
	connect:CreateVocabulary
	connect:CreateWorkspace
	connect:DeactivateEvaluationForm
	connect:DeleteContactEvaluation
	connect:DeleteContactFlow
	connect:DeleteContactFlowModule
	connect:DeleteContactFlowModuleAlias

服务前缀	操作
	connect:DeleteContactFlowModuleVersion
	connect:DeleteContactFlowVersion
	connect:DeleteDataTable
	connect:DeleteDataTableAttribute
	connect:DeleteEmailAddress
	connect:DeleteEvaluationForm
	connect:DeleteHoursOfOperation
	connect:DeleteHoursOfOperationOverride
	connect:DeleteInstance
	connect:DeleteIntegrationAssociation
	connect:DeletePredefinedAttribute
	connect:DeletePrompt
	connect:DeletePushNotificationRegistration
	connect:DeleteQueue
	connect:DeleteQuickConnect
	connect:DeleteRoutingProfile
	connect:DeleteRule
	connect:DeleteSecurityProfile
	connect:DeleteTaskTemplate
	connect:DeleteTrafficDistributionGroup
	connect:DeleteUseCase

服务前缀	操作
	connect:DeleteUser
	connect:DeleteUserHierarchyGroup
	connect:DeleteView
	connect:DeleteVocabulary
	connect:DeleteWorkspace
	connect:DeleteWorkspaceMedia
	connect:DescribeAuthenticationProfile
	connect:DescribeContactFlowModuleAlias
	connect:DescribeDataTableAttribute
	connect:DescribeHoursOfOperationOverride
	connect:DescribeInstanceAttribute
	connect:DescribeInstanceStorageConfig
	connect:DescribePhoneNumber
	connect:DescribeRule
	connect:DescribeTrafficDistributionGroup
	connect:DescribeUserHierarchyStructure
	connect:DescribeVocabulary
	connect:DisassociateAnalyticsDataSet
	connect:DisassociateApprovedOrigin
	connect:DisassociateBot
	connect:DisassociateEmailAddressAlias

服务前缀	操作
	connect:DisassociateFlow
	connect:DisassociateInstanceStorageConfig
	connect:DisassociateLambdaFunction
	connect:DisassociateLexBot
	connect:DisassociatePhoneNumberContactFlow
	connect:DisassociateQueueQuickConnects
	connect:DisassociateRoutingProfileQueues
	connect:DisassociateSecurityKey
	connect:DisassociateUserProficiencies
	connect:DismissUserContact
	connect:EvaluateDataTableValues
	connect:GetContactAttributes
	connect:GetContactMetrics
	connect:GetCurrentMetricData
	connect:GetCurrentUserData
	connect:GetEffectiveHoursOfOperations
	connect:GetFederationToken
	connect:GetFlowAssociation
	connect:GetMetricData
	connect:GetMetricDataV2
	connect:GetPromptFile

服务前缀	操作
	connect:GetTaskTemplate
	connect:GetTrafficDistribution
	connect:ImportPhoneNumber
	connect:ImportWorkspaceMedia
	connect:ListAnalyticsDataAssociations
	connect:ListAnalyticsDataLakeDataSets
	connect:ListApprovedOrigins
	connect:ListAssociatedContacts
	connect:ListAuthenticationProfiles
	connect:ListBots
	connect:ListContactEvaluations
	connect:ListContactFlowModuleAliases
	connect:ListContactFlowModuleVersions
	connect:ListContactFlowModules
	connect:ListContactFlowVersions
	connect:ListContactFlows
	connect:ListContactReferences
	connect:ListDataTableAttributes
	connect:ListDataTablePrimaryValues
	connect:ListDataTableValues
	connect:ListDataTables

服务前缀	操作
	connect:ListDefaultVocabularies
	connect:ListEvaluationFormVersions
	connect:ListEvaluationForms
	connect:ListFlowAssociations
	connect:ListHoursOfOperations
	connect:ListInstanceAttributes
	connect:ListInstanceStorageConfigs
	connect:ListIntegrationAssociations
	connect:ListLambdaFunctions
	connect:ListLexBots
	connect:ListPhoneNumbers
	connect:ListPhoneNumbersV2
	connect:ListPredefinedAttributes
	connect:ListPrompts
	connect:ListQueueQuickConnects
	connect:ListQueues
	connect:ListQuickConnects
	connect:ListRealtimeContactAnalysisSegmentsV2
	connect:ListRoutingProfileManualAssignmentQueues
	connect:ListRoutingProfileQueues
	connect:ListRoutingProfiles

服务前缀	操作
	connect:ListRules
	connect:ListSecurityKeys
	connect:ListSecurityProfileApplications
	connect:ListSecurityProfileFlowModules
	connect:ListSecurityProfilePermissions
	connect:ListSecurityProfiles
	connect:ListTaskTemplates
	connect:ListTrafficDistributionGroups
	connect:ListUseCases
	connect:ListUserHierarchyGroups
	connect:ListUsers
	connect:ListViewVersions
	connect:ListViews
	connect:ListWorkspaceMedia
	connect:ListWorkspacePages
	connect:ListWorkspaces
	connect:MonitorContact
	connect:PauseContact
	connect:PutUserStatus
	connect:ReleasePhoneNumber
	connect:ReplicateInstance

服务前缀	操作
	connect:ResumeContact
	connect:ResumeContactRecording
	connect:SearchAgentStatuses
	connect:SearchAvailablePhoneNumbers
	connect:SearchContactEvaluations
	connect:SearchContactFlowModules
	connect:SearchContactFlows
	connect:SearchContacts
	connect:SearchDataTables
	connect:SearchEmailAddresses
	connect:SearchEvaluationForms
	connect:SearchHoursOfOperations
	connect:SearchPredefinedAttributes
	connect:SearchPrompts
	connect:SearchQueues
	connect:SearchQuickConnects
	connect:SearchRoutingProfiles
	connect:SearchSecurityProfiles
	connect:SearchUserHierarchyGroups
	connect:SearchViews
	connect:SearchVocabularies

服务前缀	操作
	connect:SearchWorkspaceAssociations
	connect:SearchWorkspaces
	connect:SendChatIntegrationEvent
	connect:SendOutboundEmail
	connect:StartChatContact
	connect:StartContactEvaluation
	connect:StartContactMediaProcessing
	connect:StartContactRecording
	connect:StartContactStreaming
	connect:StartEmailContact
	connect:StartOutboundChatContact
	connect:StartOutboundEmailContact
	connect:StartOutboundVoiceContact
	connect:StartScreenSharing
	connect:StartTaskContact
	connect:StartWebRTCContact
	connect:StopContact
	connect:StopContactMediaProcessing
	connect:StopContactRecording
	connect:StopContactStreaming
	connect:SubmitContactEvaluation

服务前缀	操作
	connect:SuspendContactRecording
	connect:TransferContact
	connect:UpdateAgentStatus
	connect:UpdateAuthenticationProfile
	connect:UpdateContact
	connect:UpdateContactAttributes
	connect:UpdateContactEvaluation
	connect:UpdateContactFlowContent
	connect:UpdateContactFlowMetadata
	connect:UpdateContactFlowModuleAlias
	connect:UpdateContactFlowModuleContent
	connect:UpdateContactFlowModuleMetadata
	connect:UpdateContactFlowName
	connect:UpdateContactRoutingData
	connect:UpdateContactSchedule
	connect:UpdateDataTableAttribute
	connect:UpdateDataTableMetadata
	connect:UpdateDataTablePrimaryValues
	connect:UpdateEmailAddressMetadata
	connect:UpdateEvaluationForm
	connect:UpdateHoursOfOperation

服务前缀	操作
	connect:UpdateHoursOfOperationOverride
	connect:UpdateInstanceAttribute
	connect:UpdateInstanceStorageConfig
	connect:UpdateParticipantAuthentication
	connect:UpdateParticipantRoleConfig
	connect:UpdatePhoneNumber
	connect:UpdatePhoneNumberMetadata
	connect:UpdatePredefinedAttribute
	connect:UpdatePrompt
	connect:UpdateQueueHoursOfOperation
	connect:UpdateQueueMaxContacts
	connect:UpdateQueueName
	connect:UpdateQueueOutboundCallerConfig
	connect:UpdateQueueOutboundEmailConfig
	connect:UpdateQueueStatus
	connect:UpdateQuickConnectConfig
	connect:UpdateQuickConnectName
	connect:UpdateRoutingProfileAgentAvailabilityTimer
	connect:UpdateRoutingProfileConcurrency
	connect:UpdateRoutingProfileDefaultOutboundQueue
	connect:UpdateRoutingProfileName

服务前缀	操作
	connect:UpdateRoutingProfileQueues
	connect:UpdateRule
	connect:UpdateSecurityProfile
	connect:UpdateTaskTemplate
	connect:UpdateTrafficDistribution
	connect:UpdateUserHierarchy
	connect:UpdateUserHierarchyGroupName
	connect:UpdateUserHierarchyStructure
	connect:UpdateUserIdentityInfo
	connect:UpdateUserPhoneConfig
	connect:UpdateUserProficiencies
	connect:UpdateUserRoutingProfile
	connect:UpdateUserSecurityProfiles
	connect:UpdateViewContent
	connect:UpdateViewMetadata
	connect:UpdateWorkspaceMetadata
	connect:UpdateWorkspaceTheme
	connect:UpdateWorkspaceVisibility

服务前缀	操作
cur	cur:DeleteReportDefinition cur:DescribeReportDefinitions cur:ModifyReportDefinition cur:PutReportDefinition

服务前缀	操作
databrew	databrew:BatchDeleteRecipeVersion
	databrew:CreateDataset
	databrew:CreateProfileJob
	databrew:CreateProject
	databrew:CreateRecipe
	databrew:CreateRecipeJob
	databrew:CreateRuleset
	databrew:CreateSchedule
	databrew:DeleteDataset
	databrew:DeleteJob
	databrew:DeleteProject
	databrew:DeleteRecipeVersion
	databrew:DeleteRuleset
	databrew:DeleteSchedule
	databrew:DescribeDataset
	databrew:DescribeJob
	databrew:DescribeJobRun
	databrew:DescribeProject
	databrew:DescribeRecipe
	databrew:DescribeRuleset
	databrew:DescribeSchedule

服务前缀	操作
	databrew:ListDatasets
	databrew:ListJobRuns
	databrew:ListJobs
	databrew:ListProjects
	databrew:ListRecipeVersions
	databrew:ListRecipes
	databrew:ListRulesets
	databrew:ListSchedules
	databrew:PublishRecipe
	databrew:SendProjectSessionAction
	databrew:StartJobRun
	databrew:StartProjectSession
	databrew:StopJobRun
	databrew:UpdateDataset
	databrew:UpdateProfileJob
	databrew:UpdateProject
	databrew:UpdateRecipe
	databrew:UpdateRecipeJob
	databrew:UpdateRuleset
	databrew:UpdateSchedule

服务前缀	操作
dataexchange	dataexchange:AcceptDataGrant dataexchange:CancelJob dataexchange:CreateDataGrant dataexchange:CreateDataSet dataexchange:CreateEventAction dataexchange:CreateJob dataexchange:CreateRevision dataexchange:DeleteAsset dataexchange:DeleteDataGrant dataexchange:DeleteEventAction dataexchange:DeleteRevision dataexchange:GetDataGrant dataexchange:GetEventAction dataexchange:GetJob dataexchange:GetReceivedDataGrant dataexchange:ListDataGrants dataexchange:ListDataSetRevisions dataexchange:ListDataSets dataexchange:ListEventActions dataexchange:ListJobs dataexchange:ListReceivedDataGrants

服务前缀	操作
	dataexchange:ListRevisionAssets dataexchange:RevokeRevision dataexchange:SendDataSetNotification dataexchange:StartJob dataexchange:UpdateAsset dataexchange:UpdateDataSet dataexchange:UpdateEventAction dataexchange:UpdateRevision

服务前缀	操作
datapipeline	datapipeline:ActivatePipeline datapipeline:CreatePipeline datapipeline:DeactivatePipeline datapipeline:DeletePipeline datapipeline:DescribeObjects datapipeline:DescribePipelines datapipeline:EvaluateExpression datapipeline:GetPipelineDefinition datapipeline:ListPipelines datapipeline:PollForTask datapipeline:PutPipelineDefinition datapipeline:QueryObjects datapipeline:ReportTaskProgress datapipeline:ReportTaskRunnerHeartbeat datapipeline:SetStatus datapipeline:SetTaskStatus datapipeline:ValidatePipelineDefinition

服务前缀	操作
dax	dax:CreateCluster dax:DecreaseReplicationFactor dax:DeleteCluster dax:DeleteParameterGroup dax:DeleteSubnetGroup dax:DescribeClusters dax:DescribeDefaultParameters dax:DescribeEvents dax:DescribeParameterGroups dax:DescribeParameters dax:DescribeSubnetGroups dax:IncreaseReplicationFactor dax:RebootNode dax:UpdateCluster dax:UpdateParameterGroup dax:UpdateSubnetGroup

服务前缀	操作
devicefarm	devicefarm:CreateDevicePool
	devicefarm:CreateInstanceProfile
	devicefarm:CreateNetworkProfile
	devicefarm:CreateProject
	devicefarm:CreateRemoteAccessSession
	devicefarm:CreateTestGridProject
	devicefarm:CreateTestGridUrl
	devicefarm:CreateUpload
	devicefarm:CreateVPCEConfiguration
	devicefarm:DeleteDevicePool
	devicefarm:DeleteInstanceProfile
	devicefarm:DeleteNetworkProfile
	devicefarm:DeleteProject
	devicefarm:DeleteRemoteAccessSession
	devicefarm:DeleteRun
	devicefarm:DeleteTestGridProject
	devicefarm:DeleteUpload
	devicefarm:DeleteVPCEConfiguration
	devicefarm:GetAccountSettings
	devicefarm:GetDevice
	devicefarm:GetDeviceInstance

服务前缀	操作
	devicefarm:GetDevicePool
	devicefarm:GetDevicePoolCompatibility
	devicefarm:GetInstanceProfile
	devicefarm:GetJob
	devicefarm:GetNetworkProfile
	devicefarm:GetOfferingStatus
	devicefarm:GetProject
	devicefarm:GetRemoteAccessSession
	devicefarm:GetRun
	devicefarm:GetSuite
	devicefarm:GetTest
	devicefarm:GetTestGridProject
	devicefarm:GetTestGridSession
	devicefarm:GetUpload
	devicefarm:GetVPCEConfiguration
	devicefarm:ListArtifacts
	devicefarm:ListDeviceInstances
	devicefarm:ListDevicePools
	devicefarm:ListDevices
	devicefarm:ListInstanceProfiles
	devicefarm:ListJobs

服务前缀	操作
	devicefarm:ListNetworkProfiles
	devicefarm:ListOfferingPromotions
	devicefarm:ListOfferingTransactions
	devicefarm:ListOfferings
	devicefarm:ListProjects
	devicefarm:ListRemoteAccessSessions
	devicefarm:ListRuns
	devicefarm:ListSamples
	devicefarm:ListSuites
	devicefarm:ListTestGridProjects
	devicefarm:ListTestGridSessionActions
	devicefarm:ListTestGridSessionArtifacts
	devicefarm:ListTestGridSessions
	devicefarm:ListTests
	devicefarm:ListUniqueProblems
	devicefarm:ListUploads
	devicefarm:ListVPCEConfigurations
	devicefarm:PurchaseOffering
	devicefarm:RenewOffering
	devicefarm:ScheduleRun
	devicefarm:StopJob

服务前缀	操作
	devicefarm:StopRemoteAccessSession
	devicefarm:StopRun
	devicefarm:UpdateDeviceInstance
	devicefarm:UpdateDevicePool
	devicefarm:UpdateInstanceProfile
	devicefarm:UpdateNetworkProfile
	devicefarm:UpdateProject
	devicefarm:UpdateTestGridProject
	devicefarm:UpdateUpload
	devicefarm:UpdateVPCEConfiguration

服务前缀	操作
devops-guru	devops-guru:AddNotificationChannel devops-guru:DeleteInsight devops-guru:DescribeAccountHealth devops-guru:DescribeAccountOverview devops-guru:DescribeAnomaly devops-guru:DescribeEventSourcesConfig devops-guru:DescribeFeedback devops-guru:DescribeInsight devops-guru:DescribeOrganizationHealth devops-guru:DescribeOrganizationOverview devops-guru:DescribeOrganizationResourceCollectionHealth devops-guru:DescribeResourceCollectionHealth devops-guru:DescribeServiceIntegration devops-guru:GetCostEstimation devops-guru:GetResourceCollection devops-guru:ListAnomaliesForInsight devops-guru:ListAnomalousLogGroups devops-guru:ListEvents devops-guru:ListInsights devops-guru:ListMonitoredResources devops-guru:ListNotificationChannels

服务前缀	操作
	devops-guru:ListOrganizationInsights devops-guru:ListRecommendations devops-guru:PutFeedback devops-guru:RemoveNotificationChannel devops-guru:SearchInsights devops-guru:SearchOrganizationInsights devops-guru:StartCostEstimation devops-guru:UpdateEventSourcesConfig devops-guru:UpdateResourceCollection devops-guru:UpdateServiceIntegration

服务前缀	操作
directconnect	directconnect:AcceptDirectConnectGatewayAssociationProposal
	directconnect:AllocateConnectionOnInterconnect
	directconnect:AllocateHostedConnection
	directconnect:AllocatePrivateVirtualInterface
	directconnect:AllocatePublicVirtualInterface
	directconnect:AllocateTransitVirtualInterface
	directconnect:AssociateConnectionWithLag
	directconnect:AssociateHostedConnection
	directconnect:AssociateMacSecKey
	directconnect:AssociateVirtualInterface
	directconnect:ConfirmConnection
	directconnect:ConfirmCustomerAgreement
	directconnect:ConfirmPrivateVirtualInterface
	directconnect:ConfirmPublicVirtualInterface
	directconnect:ConfirmTransitVirtualInterface
	directconnect:CreateBGPPeer
	directconnect:CreateConnection
	directconnect:CreateDirectConnectGateway
	directconnect:CreateDirectConnectGatewayAssociation
	directconnect:CreateDirectConnectGatewayAssociationProposal
	directconnect:CreateInterconnect

服务前缀	操作
	directconnect:CreateLag
	directconnect:CreatePrivateVirtualInterface
	directconnect:CreatePublicVirtualInterface
	directconnect:CreateTransitVirtualInterface
	directconnect:DeleteBGPPeer
	directconnect:DeleteConnection
	directconnect:DeleteDirectConnectGateway
	directconnect:DeleteDirectConnectGatewayAssociation
	directconnect:DeleteDirectConnectGatewayAssociationProposal
	directconnect:DeleteInterconnect
	directconnect:DeleteLag
	directconnect:DeleteVirtualInterface
	directconnect:DescribeConnectionLoa
	directconnect:DescribeConnections
	directconnect:DescribeConnectionsOnInterconnect
	directconnect:DescribeCustomerMetadata
	directconnect:DescribeDirectConnectGatewayAssociationProposals
	directconnect:DescribeDirectConnectGatewayAssociations
	directconnect:DescribeDirectConnectGatewayAttachments
	directconnect:DescribeDirectConnectGateways
	directconnect:DescribeHostedConnections

服务前缀	操作
	directconnect:DescribeInterconnectLoa
	directconnect:DescribeInterconnects
	directconnect:DescribeLags
	directconnect:DescribeLoa
	directconnect:DescribeLocations
	directconnect:DescribeRouterConfiguration
	directconnect:DescribeVirtualGateways
	directconnect:DescribeVirtualInterfaces
	directconnect:DisassociateConnectionFromLag
	directconnect:DisassociateMacSecKey
	directconnect:ListVirtualInterfaceTestHistory
	directconnect:StartBgpFailoverTest
	directconnect:StopBgpFailoverTest
	directconnect:UpdateConnection
	directconnect:UpdateDirectConnectGateway
	directconnect:UpdateDirectConnectGatewayAssociation
	directconnect:UpdateLag
	directconnect:UpdateVirtualInterfaceAttributes

服务前缀	操作
dlm	dlm:CreateLifecyclePolicy dlm:DeleteLifecyclePolicy dlm:GetLifecyclePolicies dlm:GetLifecyclePolicy dlm:UpdateLifecyclePolicy

服务前缀	操作
dms	dms:ApplyPendingMaintenanceAction
	dms:AssociateExtensionPack
	dms:BatchStartRecommendations
	dms:CancelMetadataModelCreation
	dms:CancelReplicationTaskAssessmentRun
	dms:CreateDataProvider
	dms:CreateEndpoint
	dms:CreateEventSubscription
	dms:CreateInstanceProfile
	dms:CreateMigrationProject
	dms:CreateReplicationConfig
	dms:CreateReplicationInstance
	dms:CreateReplicationSubnetGroup
	dms:CreateReplicationTask
	dms:DeleteCertificate
	dms:DeleteConnection
	dms:DeleteDataMigration
	dms:DeleteDataProvider
	dms:DeleteEndpoint
	dms:DeleteEventSubscription
	dms:DeleteFleetAdvisorCollector

服务前缀	操作
	dms:DeleteFleetAdvisorDatabases
	dms:DeleteInstanceProfile
	dms:DeleteMigrationProject
	dms:DeleteReplicationConfig
	dms:DeleteReplicationInstance
	dms:DeleteReplicationSubnetGroup
	dms:DeleteReplicationTask
	dms:DeleteReplicationTaskAssessmentRun
	dms:DescribeAccountAttributes
	dms:DescribeApplicableIndividualAssessments
	dms:DescribeCertificates
	dms:DescribeConnections
	dms:DescribeDataMigrations
	dms:DescribeEndpointSettings
	dms:DescribeEndpointTypes
	dms:DescribeEndpoints
	dms:DescribeEngineVersions
	dms:DescribeEventCategories
	dms:DescribeEventSubscriptions
	dms:DescribeEvents
	dms:DescribeFleetAdvisorCollectors

服务前缀	操作
	dms:DescribeFleetAdvisorDatabases
	dms:DescribeFleetAdvisorLsaAnalysis
	dms:DescribeFleetAdvisorSchemaObjectSummary
	dms:DescribeFleetAdvisorSchemas
	dms:DescribeMetadataModel
	dms:DescribeMetadataModelChildren
	dms:DescribeMetadataModelCreations
	dms:DescribeMetadataModelImports
	dms:DescribeOrderableReplicationInstances
	dms:DescribePendingMaintenanceActions
	dms:DescribeRecommendationLimitations
	dms:DescribeRecommendations
	dms:DescribeRefreshSchemasStatus
	dms:DescribeReplicationConfigs
	dms:DescribeReplicationInstanceTaskLogs
	dms:DescribeReplicationInstances
	dms:DescribeReplicationSubnetGroups
	dms:DescribeReplicationTableStatistics
	dms:DescribeReplicationTaskAssessmentResults
	dms:DescribeReplicationTaskAssessmentRuns
	dms:DescribeReplicationTaskIndividualAssessments

服务前缀	操作
	dms:DescribeReplicationTasks
	dms:DescribeReplications
	dms:DescribeSchemas
	dms:DescribeTableStatistics
	dms:ExportMetadataModelAssessment
	dms:ImportCertificate
	dms:ListDataProviders
	dms:ListExtensionPacks
	dms:ListInstanceProfiles
	dms:ListMetadataModelAssessments
	dms:ListMetadataModelConversions
	dms:ListMetadataModelExports
	dms:ListMigrationProjects
	dms:ModifyDataMigration
	dms:ModifyEndpoint
	dms:ModifyEventSubscription
	dms:ModifyReplicationConfig
	dms:ModifyReplicationInstance
	dms:ModifyReplicationSubnetGroup
	dms:ModifyReplicationTask
	dms:MoveReplicationTask

服务前缀	操作
	dms:RebootReplicationInstance
	dms:RefreshSchemas
	dms:ReloadReplicationTables
	dms:ReloadTables
	dms:RunFleetAdvisorLsaAnalysis
	dms:StartMetadataModelAssessment
	dms:StartMetadataModelConversion
	dms:StartMetadataModelCreation
	dms:StartMetadataModelExportAsScripts
	dms:StartMetadataModelExportToTarget
	dms:StartRecommendations
	dms:StartReplication
	dms:StartReplicationTask
	dms:StartReplicationTaskAssessment
	dms:StopDataMigration
	dms:StopReplicationTask
	dms:TestConnection
	dms:UpdateConversionConfiguration
	dms:UpdateDataProvider
	dms:UpdateInstanceProfile
	dms:UpdateMigrationProject

服务前缀	操作
	dms:UpdateSubscriptionsToEventBridge
docdb-elastic	docdb-elastic:ApplyPendingMaintenanceAction docdb-elastic:CopyClusterSnapshot docdb-elastic:DeleteCluster docdb-elastic:DeleteClusterSnapshot docdb-elastic:GetCluster docdb-elastic:GetClusterSnapshot docdb-elastic:GetPendingMaintenanceAction docdb-elastic:ListClusterSnapshots docdb-elastic:ListClusters docdb-elastic:ListPendingMaintenanceActions docdb-elastic:RestoreClusterFromSnapshot docdb-elastic:StartCluster docdb-elastic:StopCluster docdb-elastic:UpdateCluster

服务前缀	操作
dynamodb	dynamodb:CreateBackup
	dynamodb:CreateGlobalTable
	dynamodb:CreateTable
	dynamodb:DeleteBackup
	dynamodb:DeleteTable
	dynamodb:DescribeBackup
	dynamodb:DescribeContinuousBackups
	dynamodb:DescribeContributorInsights
	dynamodb:DescribeEndpoints
	dynamodb:DescribeExport
	dynamodb:DescribeGlobalTable
	dynamodb:DescribeGlobalTableSettings
	dynamodb:DescribeImport
	dynamodb:DescribeKinesisStreamingDestination
	dynamodb:DescribeLimits
	dynamodb:DescribeStream
	dynamodb:DescribeTable
	dynamodb:DescribeTableReplicaAutoScaling
	dynamodb:DescribeTimeToLive
	dynamodb:DisableKinesisStreamingDestination
	dynamodb:EnableKinesisStreamingDestination

服务前缀	操作
	dynamodb:ExportTableToPointInTime
	dynamodb:GetResourcePolicy
	dynamodb:ImportTable
	dynamodb:ListBackups
	dynamodb:ListContributorInsights
	dynamodb:ListExports
	dynamodb:ListGlobalTables
	dynamodb:ListImports
	dynamodb:ListStreams
	dynamodb:ListTables
	dynamodb:RestoreTableFromBackup
	dynamodb:RestoreTableToPointInTime
	dynamodb:UpdateContinuousBackups
	dynamodb:UpdateContributorInsights
	dynamodb:UpdateGlobalTable
	dynamodb:UpdateGlobalTableSettings
	dynamodb:UpdateKinesisStreamingDestination
	dynamodb:UpdateTable
	dynamodb:UpdateTableReplicaAutoScaling
	dynamodb:UpdateTimeToLive

服务前缀	操作
ebs	ebs:CompleteSnapshot ebs:StartSnapshot

服务前缀	操作
ec2	ec2:AcceptAddressTransfer
	ec2:AcceptCapacityReservationBillingOwnership
	ec2:AcceptReservedInstancesExchangeQuote
	ec2:AcceptTransitGatewayMulticastDomainAssociations
	ec2:AcceptTransitGatewayPeeringAttachment
	ec2:AcceptTransitGatewayVpcAttachment
	ec2:AcceptVpcEndpointConnections
	ec2:AcceptVpcPeeringConnection
	ec2:AdvertiseByoipCidr
	ec2:AllocateAddress
	ec2:AllocateHosts
	ec2:AllocateIpamPoolCidr
	ec2:ApplySecurityGroupsToClientVpnTargetNetwork
	ec2:AssignIpv6Addresses
	ec2:AssignPrivateIpAddresses
	ec2:AssignPrivateNatGatewayAddress
	ec2:AssociateAddress
	ec2:AssociateCapacityReservationBillingOwner
	ec2:AssociateClientVpnTargetNetwork
	ec2:AssociateDhcpOptions
	ec2:AssociateEnclaveCertificateIamRole

服务前缀	操作
	ec2:AssociateIamInstanceProfile
	ec2:AssociateInstanceEventWindow
	ec2:AssociateIpamByoasn
	ec2:AssociateIpamResourceDiscovery
	ec2:AssociateNatGatewayAddress
	ec2:AssociateRouteServer
	ec2:AssociateRouteTable
	ec2:AssociateSecurityGroupVpc
	ec2:AssociateSubnetCidrBlock
	ec2:AssociateTransitGatewayMulticastDomain
	ec2:AssociateTransitGatewayPolicyTable
	ec2:AssociateTransitGatewayRouteTable
	ec2:AssociateTrunkInterface
	ec2:AssociateVpcCidrBlock
	ec2:AttachClassicLinkVpc
	ec2:AttachInternetGateway
	ec2:AttachNetworkInterface
	ec2:AttachVerifiedAccessTrustProvider
	ec2:AttachVolume
	ec2:AttachVpnGateway
	ec2:AuthorizeClientVpnIngress

服务前缀	操作
	ec2:AuthorizeSecurityGroupEgress
	ec2:AuthorizeSecurityGroupIngress
	ec2:BundleInstance
	ec2:CancelBundleTask
	ec2:CancelCapacityReservation
	ec2:CancelCapacityReservationFleets
	ec2:CancelConversionTask
	ec2:CancelDeclarativePoliciesReport
	ec2:CancelExportTask
	ec2:CancelImageLaunchPermission
	ec2:CancelImportTask
	ec2:CancelReservedInstancesListing
	ec2:CancelSpotFleetRequests
	ec2:CancelSpotInstanceRequests
	ec2:ConfirmProductInstance
	ec2:CopyFpgaImage
	ec2:CopyImage
	ec2:CopySnapshot
	ec2:CopyVolumes
	ec2:CreateCapacityManagerDataExport
	ec2:CreateCapacityReservation

服务前缀	操作
	ec2:CreateCapacityReservationBySplitting
	ec2:CreateCapacityReservationFleet
	ec2:CreateCarrierGateway
	ec2:CreateClientVpnEndpoint
	ec2:CreateClientVpnRoute
	ec2:CreateCoipCidr
	ec2:CreateCoipPool
	ec2:CreateCustomerGateway
	ec2:CreateDefaultSubnet
	ec2:CreateDefaultVpc
	ec2:CreateDelegateMacVolumeOwnershipTask
	ec2:CreateDhcpOptions
	ec2:CreateEgressOnlyInternetGateway
	ec2:CreateFleet
	ec2:CreateFlowLogs
	ec2:CreateFpgaImage
	ec2:CreateImage
	ec2:CreateImageUsageReport
	ec2:CreateInstanceConnectEndpoint
	ec2:CreateInstanceEventWindow
	ec2:CreateInstanceExportTask

服务前缀	操作
	ec2:CreateInternetGateway
	ec2:CreateInterruptibleCapacityReservationAllocation
	ec2:CreateIpam
	ec2:CreateIpamExternalResourceVerificationToken
	ec2:CreateIpamPolicy
	ec2:CreateIpamPool
	ec2:CreateIpamPrefixListResolver
	ec2:CreateIpamPrefixListResolverTarget
	ec2:CreateIpamResourceDiscovery
	ec2:CreateIpamScope
	ec2:CreateKeyPair
	ec2:CreateLaunchTemplateVersion
	ec2:CreateLocalGatewayRoute
	ec2:CreateLocalGatewayRouteTable
	ec2:CreateLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:CreateLocalGatewayRouteTableVpcAssociation
	ec2:CreateLocalGatewayVirtualInterface
	ec2:CreateLocalGatewayVirtualInterfaceGroup
	ec2:CreateMacSystemIntegrityProtectionModificationTask
	ec2:CreateManagedPrefixList

服务前缀	操作
	ec2:CreateNatGateway
	ec2:CreateNetworkAcl
	ec2:CreateNetworkAclEntry
	ec2:CreateNetworkInsightsAccessScope
	ec2:CreateNetworkInsightsPath
	ec2:CreateNetworkInterface
	ec2:CreateNetworkInterfacePermission
	ec2:CreatePlacementGroup
	ec2:CreatePublicIpv4Pool
	ec2:CreateReplaceRootVolumeTask
	ec2:CreateReservedInstancesListing
	ec2:CreateRestoreImageTask
	ec2:CreateRoute
	ec2:CreateRouteServer
	ec2:CreateRouteServerEndpoint
	ec2:CreateRouteServerPeer
	ec2:CreateRouteTable
	ec2:CreateSecurityGroup
	ec2:CreateSnapshots
	ec2:CreateSpotDatafeedSubscription
	ec2:CreateStoreImageTask

服务前缀	操作
	ec2:CreateSubnet
	ec2:CreateSubnetCidrReservation
	ec2:CreateTrafficMirrorFilter
	ec2:CreateTrafficMirrorFilterRule
	ec2:CreateTrafficMirrorSession
	ec2:CreateTrafficMirrorTarget
	ec2:CreateTransitGateway
	ec2:CreateTransitGatewayConnect
	ec2:CreateTransitGatewayConnectPeer
	ec2:CreateTransitGatewayMeteringPolicy
	ec2:CreateTransitGatewayMeteringPolicyEntry
	ec2:CreateTransitGatewayMulticastDomain
	ec2:CreateTransitGatewayPeeringAttachment
	ec2:CreateTransitGatewayPolicyTable
	ec2:CreateTransitGatewayPrefixListReference
	ec2:CreateTransitGatewayRoute
	ec2:CreateTransitGatewayRouteTable
	ec2:CreateTransitGatewayRouteTableAnnouncement
	ec2:CreateTransitGatewayVpcAttachment
	ec2:CreateVerifiedAccessEndpoint
	ec2:CreateVerifiedAccessGroup

服务前缀	操作
	ec2:CreateVerifiedAccessInstance
	ec2:CreateVerifiedAccessTrustProvider
	ec2:CreateVolume
	ec2:CreateVpc
	ec2:CreateVpcBlockPublicAccessExclusion
	ec2:CreateVpcEncryptionControl
	ec2:CreateVpcEndpoint
	ec2:CreateVpcEndpointConnectionNotification
	ec2:CreateVpcEndpointServiceConfiguration
	ec2:CreateVpcPeeringConnection
	ec2:CreateVpnConcentrator
	ec2:CreateVpnConnection
	ec2:CreateVpnConnectionRoute
	ec2:CreateVpnGateway
	ec2:DeleteCapacityManagerDataExport
	ec2:DeleteCarrierGateway
	ec2:DeleteClientVpnEndpoint
	ec2:DeleteClientVpnRoute
	ec2:DeleteCoipCidr
	ec2:DeleteCoipPool
	ec2:DeleteCustomerGateway

服务前缀	操作
	ec2:DeleteDhcpOptions
	ec2:DeleteEgressOnlyInternetGateway
	ec2:DeleteFleets
	ec2:DeleteFlowLogs
	ec2:DeleteFpgaImage
	ec2:DeleteImageUsageReport
	ec2:DeleteInstanceConnectEndpoint
	ec2:DeleteInstanceEventWindow
	ec2:DeleteInternetGateway
	ec2:DeleteIpam
	ec2:DeleteIpamExternalResourceVerificationToken
	ec2:DeleteIpamPolicy
	ec2:DeleteIpamPool
	ec2:DeleteIpamPrefixListResolver
	ec2:DeleteIpamPrefixListResolverTarget
	ec2:DeleteIpamResourceDiscovery
	ec2:DeleteIpamScope
	ec2:DeleteKeyPair
	ec2:DeleteLaunchTemplate
	ec2:DeleteLaunchTemplateVersions
	ec2:DeleteLocalGatewayRoute

服务前缀	操作
	ec2:DeleteLocalGatewayRouteTable
	ec2:DeleteLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:DeleteLocalGatewayRouteTableVpcAssociation
	ec2:DeleteLocalGatewayVirtualInterface
	ec2:DeleteLocalGatewayVirtualInterfaceGroup
	ec2:DeleteManagedPrefixList
	ec2:DeleteNatGateway
	ec2:DeleteNetworkAcl
	ec2:DeleteNetworkAclEntry
	ec2:DeleteNetworkInsightsAccessScope
	ec2:DeleteNetworkInsightsAccessScopeAnalysis
	ec2:DeleteNetworkInsightsAnalysis
	ec2:DeleteNetworkInsightsPath
	ec2:DeleteNetworkInterface
	ec2:DeleteNetworkInterfacePermission
	ec2:DeletePlacementGroup
	ec2:DeletePublicIpv4Pool
	ec2:DeleteQueuedReservedInstances
	ec2:DeleteRoute
	ec2:DeleteRouteServer

服务前缀	操作
	ec2:DeleteRouteServerEndpoint
	ec2:DeleteRouteServerPeer
	ec2:DeleteRouteTable
	ec2:DeleteSecurityGroup
	ec2:DeleteSpotDatafeedSubscription
	ec2:DeleteSubnet
	ec2:DeleteSubnetCidrReservation
	ec2:DeleteTrafficMirrorFilter
	ec2:DeleteTrafficMirrorFilterRule
	ec2:DeleteTrafficMirrorSession
	ec2:DeleteTrafficMirrorTarget
	ec2:DeleteTransitGateway
	ec2:DeleteTransitGatewayConnect
	ec2:DeleteTransitGatewayConnectPeer
	ec2:DeleteTransitGatewayMeteringPolicy
	ec2:DeleteTransitGatewayMeteringPolicyEntry
	ec2:DeleteTransitGatewayMulticastDomain
	ec2:DeleteTransitGatewayPeeringAttachment
	ec2:DeleteTransitGatewayPolicyTable
	ec2:DeleteTransitGatewayPrefixListReference
	ec2:DeleteTransitGatewayRoute

服务前缀	操作
	ec2:DeleteTransitGatewayRouteTable
	ec2:DeleteTransitGatewayRouteTableAnnouncement
	ec2:DeleteTransitGatewayVpcAttachment
	ec2:DeleteVerifiedAccessEndpoint
	ec2:DeleteVerifiedAccessGroup
	ec2:DeleteVerifiedAccessInstance
	ec2:DeleteVerifiedAccessTrustProvider
	ec2:DeleteVolume
	ec2:DeleteVpc
	ec2:DeleteVpcBlockPublicAccessExclusion
	ec2:DeleteVpcEncryptionControl
	ec2:DeleteVpcEndpointConnectionNotifications
	ec2:DeleteVpcEndpointServiceConfigurations
	ec2:DeleteVpcEndpoints
	ec2:DeleteVpcPeeringConnection
	ec2:DeleteVpnConcentrator
	ec2:DeleteVpnConnection
	ec2:DeleteVpnConnectionRoute
	ec2:DeleteVpnGateway
	ec2:DeprovisionByoipCidr
	ec2:DeprovisionIpamByoasn

服务前缀	操作
	ec2:DeprovisionIpamPoolCidr
	ec2:DeprovisionPublicIpv4PoolCidr
	ec2:DeregisterImage
	ec2:DeregisterInstanceEventNotificationAttributes
	ec2:DeregisterTransitGatewayMulticastGroupMembers
	ec2:DeregisterTransitGatewayMulticastGroupSources
	ec2:DescribeAccountAttributes
	ec2:DescribeAddressTransfers
	ec2:DescribeAddresses
	ec2:DescribeAddressesAttribute
	ec2:DescribeAggregateIdFormat
	ec2:DescribeAvailabilityZones
	ec2:DescribeAwsNetworkPerformanceMetricSubscriptions
	ec2:DescribeBundleTasks
	ec2:DescribeByoipCidrs
	ec2:DescribeCapacityBlockExtensionHistory
	ec2:DescribeCapacityBlockExtensionOfferings
	ec2:DescribeCapacityBlockStatus
	ec2:DescribeCapacityBlocks
	ec2:DescribeCapacityManagerDataExports
	ec2:DescribeCapacityReservationBillingRequests

服务前缀	操作
	ec2:DescribeCapacityReservationFleets
	ec2:DescribeCapacityReservationTopology
	ec2:DescribeCapacityReservations
	ec2:DescribeCarrierGateways
	ec2:DescribeClassicLinkInstances
	ec2:DescribeClientVpnAuthorizationRules
	ec2:DescribeClientVpnConnections
	ec2:DescribeClientVpnEndpoints
	ec2:DescribeClientVpnRoutes
	ec2:DescribeClientVpnTargetNetworks
	ec2:DescribeCoipPools
	ec2:DescribeConversionTasks
	ec2:DescribeCustomerGateways
	ec2:DescribeDeclarativePoliciesReports
	ec2:DescribeDhcpOptions
	ec2:DescribeEgressOnlyInternetGateways
	ec2:DescribeElasticGpus
	ec2:DescribeExportImageTasks
	ec2:DescribeExportTasks
	ec2:DescribeFastLaunchImages
	ec2:DescribeFastSnapshotRestores

服务前缀	操作
	ec2:DescribeFleetHistory
	ec2:DescribeFleetInstances
	ec2:DescribeFleets
	ec2:DescribeFlowLogs
	ec2:DescribeFpgaImageAttribute
	ec2:DescribeFpgaImages
	ec2:DescribeHostReservationOfferings
	ec2:DescribeHostReservations
	ec2:DescribeHosts
	ec2:DescribeIamInstanceProfileAssociations
	ec2:DescribeIdFormat
	ec2:DescribeIdentityIdFormat
	ec2:DescribeImageAttribute
	ec2:DescribeImageReferences
	ec2:DescribeImageUsageReportEntries
	ec2:DescribeImageUsageReports
	ec2:DescribeImportImageTasks
	ec2:DescribeImportSnapshotTasks
	ec2:DescribeInstanceConnectEndpoints
	ec2:DescribeInstanceCreditSpecifications
	ec2:DescribeInstanceEventNotificationAttributes

服务前缀	操作
	ec2:DescribeInstanceEventWindows
	ec2:DescribeInstanceImageMetadata
	ec2:DescribeInstanceSqlHaHistoryStates
	ec2:DescribeInstanceSqlHaStates
	ec2:DescribeInstanceTopology
	ec2:DescribeInstanceTypes
	ec2:DescribeInternetGateways
	ec2:DescribeIpamByoasn
	ec2:DescribeIpamExternalResourceVerificationTokens
	ec2:DescribeIpamPolicies
	ec2:DescribeIpamPools
	ec2:DescribeIpamPrefixListResolverTargets
	ec2:DescribeIpamPrefixListResolvers
	ec2:DescribeIpamResourceDiscoveries
	ec2:DescribeIpamResourceDiscoveryAssociations
	ec2:DescribeIpamScopes
	ec2:DescribeIpams
	ec2:DescribeIpv6Pools
	ec2:DescribeKeyPairs
	ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations

服务前缀	操作
	ec2:DescribeLocalGatewayRouteTableVpcAssociations
	ec2:DescribeLocalGatewayRouteTables
	ec2:DescribeLocalGatewayVirtualInterfaceGroups
	ec2:DescribeLocalGatewayVirtualInterfaces
	ec2:DescribeLocalGateways
	ec2:DescribeLockedSnapshots
	ec2:DescribeMacHosts
	ec2:DescribeMacModificationTasks
	ec2:DescribeManagedPrefixLists
	ec2:DescribeMovingAddresses
	ec2:DescribeNatGateways
	ec2:DescribeNetworkAcls
	ec2:DescribeNetworkInsightsAccessScopeAnalyses
	ec2:DescribeNetworkInsightsAccessScopes
	ec2:DescribeNetworkInsightsAnalyses
	ec2:DescribeNetworkInsightsPaths
	ec2:DescribeNetworkInterfaceAttribute
	ec2:DescribeNetworkInterfacePermissions
	ec2:DescribeNetworkInterfaces
	ec2:DescribeOutpostLags
	ec2:DescribePlacementGroups

服务前缀	操作
	ec2:DescribePrefixLists
	ec2:DescribePrincipalIdFormat
	ec2:DescribePublicIpv4Pools
	ec2:DescribeRegions
	ec2:DescribeReplaceRootVolumeTasks
	ec2:DescribeReservedInstances
	ec2:DescribeReservedInstancesListings
	ec2:DescribeReservedInstancesModifications
	ec2:DescribeReservedInstancesOfferings
	ec2:DescribeRouteServerEndpoints
	ec2:DescribeRouteServerPeers
	ec2:DescribeRouteServers
	ec2:DescribeRouteTables
	ec2:DescribeScheduledInstanceAvailability
	ec2:DescribeScheduledInstances
	ec2:DescribeSecurityGroupReferences
	ec2:DescribeSecurityGroupRules
	ec2:DescribeSecurityGroupVpcAssociations
	ec2:DescribeSecurityGroups
	ec2:DescribeServiceLinkVirtualInterfaces
	ec2:DescribeSnapshotAttribute

服务前缀	操作
	ec2:DescribeSnapshotTierStatus
	ec2:DescribeSpotDatafeedSubscription
	ec2:DescribeSpotFleetInstances
	ec2:DescribeSpotFleetRequestHistory
	ec2:DescribeSpotFleetRequests
	ec2:DescribeSpotInstanceRequests
	ec2:DescribeSpotPriceHistory
	ec2:DescribeStaleSecurityGroups
	ec2:DescribeStoreImageTasks
	ec2:DescribeTrafficMirrorFilterRules
	ec2:DescribeTrafficMirrorFilters
	ec2:DescribeTrafficMirrorSessions
	ec2:DescribeTrafficMirrorTargets
	ec2:DescribeTransitGatewayAttachments
	ec2:DescribeTransitGatewayConnectPeers
	ec2:DescribeTransitGatewayConnects
	ec2:DescribeTransitGatewayMeteringPolicies
	ec2:DescribeTransitGatewayMulticastDomains
	ec2:DescribeTransitGatewayPeeringAttachments
	ec2:DescribeTransitGatewayPolicyTables
	ec2:DescribeTransitGatewayRouteTableAnnouncements

服务前缀	操作
	ec2:DescribeTransitGatewayRouteTables
	ec2:DescribeTransitGatewayVpcAttachments
	ec2:DescribeTransitGateways
	ec2:DescribeTrunkInterfaceAssociations
	ec2:DescribeVerifiedAccessEndpoints
	ec2:DescribeVerifiedAccessGroups
	ec2:DescribeVerifiedAccessInstanceLoggingConfigurations
	ec2:DescribeVerifiedAccessInstances
	ec2:DescribeVerifiedAccessTrustProviders
	ec2:DescribeVolumeAttribute
	ec2:DescribeVolumeStatus
	ec2:DescribeVolumes
	ec2:DescribeVolumesModifications
	ec2:DescribeVpcAttribute
	ec2:DescribeVpcBlockPublicAccessExclusions
	ec2:DescribeVpcBlockPublicAccessOptions
	ec2:DescribeVpcClassicLink
	ec2:DescribeVpcClassicLinkDnsSupport
	ec2:DescribeVpcEncryptionControls
	ec2:DescribeVpcEndpointAssociations
	ec2:DescribeVpcEndpointConnectionNotifications

服务前缀	操作
	ec2:DescribeVpcEndpointConnections
	ec2:DescribeVpcEndpointServiceConfigurations
	ec2:DescribeVpcEndpointServicePermissions
	ec2:DescribeVpcEndpointServices
	ec2:DescribeVpcEndpoints
	ec2:DescribeVpcPeeringConnections
	ec2:DescribeVpcs
	ec2:DescribeVpnConcentrators
	ec2:DescribeVpnConnections
	ec2:DescribeVpnGateways
	ec2:DetachClassicLinkVpc
	ec2:DetachInternetGateway
	ec2:DetachNetworkInterface
	ec2:DetachVerifiedAccessTrustProvider
	ec2:DetachVolume
	ec2:DetachVpnGateway
	ec2:DisableAddressTransfer
	ec2:DisableAllowedImagesSettings
	ec2:DisableAwsNetworkPerformanceMetricSubscription
	ec2:DisableCapacityManager
	ec2:DisableEbsEncryptionByDefault

服务前缀	操作
	ec2:DisableFastLaunch
	ec2:DisableFastSnapshotRestores
	ec2:DisableImage
	ec2:DisableImageBlockPublicAccess
	ec2:DisableImageDeprecation
	ec2:DisableImageDeregistrationProtection
	ec2:DisableInstanceSqlHaStandbyDetections
	ec2:DisableIpamOrganizationAdminAccount
	ec2:DisableIpamPolicy
	ec2:DisableRouteServerPropagation
	ec2:DisableSerialConsoleAccess
	ec2:DisableSnapshotBlockPublicAccess
	ec2:DisableTransitGatewayRouteTablePropagation
	ec2:DisableVgwRoutePropagation
	ec2:DisableVpcClassicLink
	ec2:DisableVpcClassicLinkDnsSupport
	ec2:DisassociateAddress
	ec2:DisassociateCapacityReservationBillingOwner
	ec2:DisassociateClientVpnTargetNetwork
	ec2:DisassociateEnclaveCertificateIamRole
	ec2:DisassociateIamInstanceProfile

服务前缀	操作
	ec2:DisassociateInstanceEventWindow
	ec2:DisassociateIpamByoasn
	ec2:DisassociateIpamResourceDiscovery
	ec2:DisassociateNatGatewayAddress
	ec2:DisassociateRouteServer
	ec2:DisassociateRouteTable
	ec2:DisassociateSecurityGroupVpc
	ec2:DisassociateSubnetCidrBlock
	ec2:DisassociateTransitGatewayMulticastDomain
	ec2:DisassociateTransitGatewayPolicyTable
	ec2:DisassociateTransitGatewayRouteTable
	ec2:DisassociateTrunkInterface
	ec2:DisassociateVpcCidrBlock
	ec2:EnableAddressTransfer
	ec2:EnableAllowedImagesSettings
	ec2:EnableAwsNetworkPerformanceMetricSubscription
	ec2:EnableCapacityManager
	ec2:EnableEbsEncryptionByDefault
	ec2:EnableFastLaunch
	ec2:EnableFastSnapshotRestores
	ec2:EnableImage

服务前缀	操作
	ec2:EnableImageBlockPublicAccess
	ec2:EnableImageDeprecation
	ec2:EnableImageDeregistrationProtection
	ec2:EnableInstanceSqlHaStandbyDetections
	ec2:EnableIpamOrganizationAdminAccount
	ec2:EnableIpamPolicy
	ec2:EnableReachabilityAnalyzerOrganizationSharing
	ec2:EnableRouteServerPropagation
	ec2:EnableSerialConsoleAccess
	ec2:EnableSnapshotBlockPublicAccess
	ec2:EnableTransitGatewayRouteTablePropagation
	ec2:EnableVgwRoutePropagation
	ec2:EnableVolumeIO
	ec2:EnableVpcClassicLink
	ec2:EnableVpcClassicLinkDnsSupport
	ec2:ExportClientVpnClientCertificateRevocationList
	ec2:ExportClientVpnClientConfiguration
	ec2:ExportImage
	ec2:ExportTransitGatewayRoutes
	ec2:ExportVerifiedAccessInstanceClientConfiguration
	ec2:GetActiveVpnTunnelStatus

服务前缀	操作
	ec2:GetAllowedImagesSettings
	ec2:GetAssociatedEnclaveCertificateIamRoles
	ec2:GetAssociatedIpv6PoolCidrs
	ec2:GetAwsNetworkPerformanceData
	ec2:GetCapacityManagerAttributes
	ec2:GetCapacityManagerMetricData
	ec2:GetCapacityManagerMetricDimensions
	ec2:GetCapacityReservationUsage
	ec2:GetCoipPoolUsage
	ec2:GetConsoleOutput
	ec2:GetConsoleScreenshot
	ec2:GetDeclarativePoliciesReportSummary
	ec2:GetDefaultCreditSpecification
	ec2:GetEbsDefaultKmsKeyId
	ec2:GetEbsEncryptionByDefault
	ec2:GetEnabledIpamPolicy
	ec2:GetFlowLogsIntegrationTemplate
	ec2:GetGroupsForCapacityReservation
	ec2:GetHostReservationPurchasePreview
	ec2:GetImageAncestry
	ec2:GetImageBlockPublicAccessState

服务前缀	操作
	ec2:GetInstanceMetadataDefaults
	ec2:GetInstanceTpmEkPub
	ec2:GetInstanceTypesFromInstanceRequirements
	ec2:GetInstanceUefiData
	ec2:GetIpamAddressHistory
	ec2:GetIpamDiscoveredAccounts
	ec2:GetIpamDiscoveredPublicAddresses
	ec2:GetIpamDiscoveredResourceCidrs
	ec2:GetIpamPolicyAllocationRules
	ec2:GetIpamPolicyOrganizationTargets
	ec2:GetIpamPoolAllocations
	ec2:GetIpamPoolCidrs
	ec2:GetIpamPrefixListResolverRules
	ec2:GetIpamPrefixListResolverVersionEntries
	ec2:GetIpamPrefixListResolverVersions
	ec2:GetIpamResourceCidrs
	ec2:GetLaunchTemplateData
	ec2:GetManagedPrefixListAssociations
	ec2:GetManagedPrefixListEntries
	ec2:GetNetworkInsightsAccessScopeAnalysisFindings
	ec2:GetNetworkInsightsAccessScopeContent

服务前缀	操作
	ec2:GetPasswordData
	ec2:GetReservedInstancesExchangeQuote
	ec2:GetRouteServerAssociations
	ec2:GetRouteServerPropagations
	ec2:GetRouteServerRoutingDatabase
	ec2:GetSecurityGroupsForVpc
	ec2:GetSerialConsoleAccessStatus
	ec2:GetSnapshotBlockPublicAccessState
	ec2:GetSpotPlacementScores
	ec2:GetSubnetCidrReservations
	ec2:GetTransitGatewayAttachmentPropagations
	ec2:GetTransitGatewayMeteringPolicyEntries
	ec2:GetTransitGatewayMulticastDomainAssociations
	ec2:GetTransitGatewayPolicyTableAssociations
	ec2:GetTransitGatewayPolicyTableEntries
	ec2:GetTransitGatewayPrefixListReferences
	ec2:GetTransitGatewayRouteTableAssociations
	ec2:GetTransitGatewayRouteTablePropagations
	ec2:GetVerifiedAccessEndpointPolicy
	ec2:GetVerifiedAccessEndpointTargets
	ec2:GetVerifiedAccessGroupPolicy

服务前缀	操作
	ec2:GetVpcResourcesBlockingEncryptionEnforcement
	ec2:GetVpnConnectionDeviceSampleConfiguration
	ec2:GetVpnConnectionDeviceTypes
	ec2:GetVpnTunnelReplacementStatus
	ec2:ImportClientVpnClientCertificateRevocationList
	ec2:ImportImage
	ec2:ImportInstance
	ec2:ImportKeyPair
	ec2:ImportSnapshot
	ec2:ImportVolume
	ec2:ListImagesInRecycleBin
	ec2:ListSnapshotsInRecycleBin
	ec2:ListVolumesInRecycleBin
	ec2:LockSnapshot
	ec2:ModifyAddressAttribute
	ec2:ModifyAvailabilityZoneGroup
	ec2:ModifyCapacityReservation
	ec2:ModifyCapacityReservationFleet
	ec2:ModifyClientVpnEndpoint
	ec2:ModifyDefaultCreditSpecification
	ec2:ModifyEbsDefaultKmsKeyId

服务前缀	操作
	ec2:ModifyFleet
	ec2:ModifyFpgaImageAttribute
	ec2:ModifyHosts
	ec2:ModifyIdFormat
	ec2:ModifyIdentityIdFormat
	ec2:ModifyImageAttribute
	ec2:ModifyInstanceAttribute
	ec2:ModifyInstanceCapacityReservationAttributes
	ec2:ModifyInstanceConnectEndpoint
	ec2:ModifyInstanceCpuOptions
	ec2:ModifyInstanceCreditSpecification
	ec2:ModifyInstanceEventStartTime
	ec2:ModifyInstanceEventWindow
	ec2:ModifyInstanceMaintenanceOptions
	ec2:ModifyInstanceMetadataDefaults
	ec2:ModifyInstanceMetadataOptions
	ec2:ModifyInstanceNetworkPerformanceOptions
	ec2:ModifyInstancePlacement
	ec2:ModifyIpam
	ec2:ModifyIpamPolicyAllocationRules
	ec2:ModifyIpamPool

服务前缀	操作
	ec2:ModifyIpamPrefixListResolver
	ec2:ModifyIpamPrefixListResolverTarget
	ec2:ModifyIpamResourceCidr
	ec2:ModifyIpamResourceDiscovery
	ec2:ModifyIpamScope
	ec2:ModifyLaunchTemplate
	ec2:ModifyLocalGatewayRoute
	ec2:ModifyManagedPrefixList
	ec2:ModifyNetworkInterfaceAttribute
	ec2:ModifyPrivateDnsNameOptions
	ec2:ModifyPublicIpDnsNameOptions
	ec2:ModifyReservedInstances
	ec2:ModifyRouteServer
	ec2:ModifySecurityGroupRules
	ec2:ModifySnapshotAttribute
	ec2:ModifySnapshotTier
	ec2:ModifySpotFleetRequest
	ec2:ModifySubnetAttribute
	ec2:ModifyTrafficMirrorFilterNetworkServices
	ec2:ModifyTrafficMirrorFilterRule
	ec2:ModifyTrafficMirrorSession

服务前缀	操作
	ec2:ModifyTransitGateway
	ec2:ModifyTransitGatewayMeteringPolicy
	ec2:ModifyTransitGatewayPrefixListReference
	ec2:ModifyTransitGatewayVpcAttachment
	ec2:ModifyVerifiedAccessEndpoint
	ec2:ModifyVerifiedAccessEndpointPolicy
	ec2:ModifyVerifiedAccessGroup
	ec2:ModifyVerifiedAccessGroupPolicy
	ec2:ModifyVerifiedAccessInstance
	ec2:ModifyVerifiedAccessInstanceLoggingConfiguration
	ec2:ModifyVerifiedAccessTrustProvider
	ec2:ModifyVolume
	ec2:ModifyVolumeAttribute
	ec2:ModifyVpcAttribute
	ec2:ModifyVpcBlockPublicAccessExclusion
	ec2:ModifyVpcBlockPublicAccessOptions
	ec2:ModifyVpcEncryptionControl
	ec2:ModifyVpcEndpoint
	ec2:ModifyVpcEndpointConnectionNotification
	ec2:ModifyVpcEndpointServiceConfiguration
	ec2:ModifyVpcEndpointServicePayerResponsibility

服务前缀	操作
	ec2:ModifyVpcEndpointServicePermissions
	ec2:ModifyVpcPeeringConnectionOptions
	ec2:ModifyVpcTenancy
	ec2:ModifyVpnConnection
	ec2:ModifyVpnConnectionOptions
	ec2:ModifyVpnTunnelCertificate
	ec2:ModifyVpnTunnelOptions
	ec2:MonitorInstances
	ec2:MoveAddressToVpc
	ec2:MoveByoipCidrToIpam
	ec2:MoveCapacityReservationInstances
	ec2:ProvisionByoipCidr
	ec2:ProvisionIpamByoasn
	ec2:ProvisionIpamPoolCidr
	ec2:ProvisionPublicIpv4PoolCidr
	ec2:PurchaseCapacityBlockExtension
	ec2:PurchaseHostReservation
	ec2:PurchaseReservedInstancesOffering
	ec2:PurchaseScheduledInstances
	ec2:RebootInstances
	ec2:RegisterImage

服务前缀	操作
	ec2:RegisterInstanceEventNotificationAttributes
	ec2:RegisterTransitGatewayMulticastGroupMembers
	ec2:RegisterTransitGatewayMulticastGroupSources
	ec2:RejectCapacityReservationBillingOwnership
	ec2:RejectTransitGatewayMulticastDomainAssociations
	ec2:RejectTransitGatewayPeeringAttachment
	ec2:RejectTransitGatewayVpcAttachment
	ec2:RejectVpcEndpointConnections
	ec2:RejectVpcPeeringConnection
	ec2:ReleaseAddress
	ec2:ReleaseHosts
	ec2:ReleaseIpamPoolAllocation
	ec2:ReplaceIamInstanceProfileAssociation
	ec2:ReplaceImageCriteriaInAllowedImagesSettings
	ec2:ReplaceNetworkAclAssociation
	ec2:ReplaceNetworkAclEntry
	ec2:ReplaceRoute
	ec2:ReplaceRouteTableAssociation
	ec2:ReplaceTransitGatewayRoute
	ec2:ReplaceVpnTunnel
	ec2:ReportInstanceStatus

服务前缀	操作
	ec2:RequestSpotFleet
	ec2:RequestSpotInstances
	ec2:ResetAddressAttribute
	ec2:ResetEbsDefaultKmsKeyId
	ec2:ResetFpgaImageAttribute
	ec2:ResetImageAttribute
	ec2:ResetInstanceAttribute
	ec2:ResetNetworkInterfaceAttribute
	ec2:ResetSnapshotAttribute
	ec2:RestoreAddressToClassic
	ec2:RestoreImageFromRecycleBin
	ec2:RestoreManagedPrefixListVersion
	ec2:RestoreSnapshotFromRecycleBin
	ec2:RestoreSnapshotTier
	ec2:RestoreVolumeFromRecycleBin
	ec2:RevokeClientVpnIngress
	ec2:RevokeSecurityGroupEgress
	ec2:RevokeSecurityGroupIngress
	ec2:RunInstances
	ec2:RunScheduledInstances
	ec2:SearchLocalGatewayRoutes

服务前缀	操作
	ec2:SearchTransitGatewayMulticastGroups
	ec2:SearchTransitGatewayRoutes
	ec2:SendDiagnosticInterrupt
	ec2:StartDeclarativePoliciesReport
	ec2:StartInstances
	ec2:StartNetworkInsightsAccessScopeAnalysis
	ec2:StartNetworkInsightsAnalysis
	ec2:StartVpcEndpointServicePrivateDnsVerification
	ec2:TerminateClientVpnConnections
	ec2:UnassignIpv6Addresses
	ec2:UnassignPrivateIpAddresses
	ec2:UnassignPrivateNatGatewayAddress
	ec2:UnlockSnapshot
	ec2:UnmonitorInstances
	ec2:UpdateCapacityManagerOrganizationsAccess
	ec2:UpdateInterruptibleCapacityReservationAllocation
	ec2:UpdateSecurityGroupRuleDescriptionsEgress
	ec2:UpdateSecurityGroupRuleDescriptionsIngress
	ec2:WithdrawByoipCidr

服务前缀	操作
ecr	ecr:BatchCheckLayerAvailability ecr:BatchDeleteImage ecr:BatchGetImage ecr:BatchGetRepositoryScanningConfiguration ecr:CompleteLayerUpload ecr:CreatePullThroughCacheRule ecr:CreateRepositoryCreationTemplate ecr:DeleteLifecyclePolicy ecr:DeletePullThroughCacheRule ecr:DeleteRegistryPolicy ecr:DeleteRepository ecr:DeleteRepositoryCreationTemplate ecr:DeleteRepositoryPolicy ecr:DeleteSigningConfiguration ecr:DescribeImageReplicationStatus ecr:DescribeImageScanFindings ecr:DescribeImages ecr:DescribePullThroughCacheRules ecr:DescribeRegistry ecr:DescribeRepositories ecr:DescribeRepositoryCreationTemplates

服务前缀	操作
	ecr:GetAccountSetting ecr:GetAuthorizationToken ecr:GetDownloadUrlForLayer ecr:GetLifecyclePolicy ecr:GetLifecyclePolicyPreview ecr:GetRegistryPolicy ecr:GetRegistryScanningConfiguration ecr:GetRepositoryPolicy ecr:GetSigningConfiguration ecr:InitiateLayerUpload ecr:ListImages ecr:ListPullTimeUpdateExclusions ecr:PutAccountSetting ecr:PutImage ecr:PutImageScanningConfiguration ecr:PutRegistryPolicy ecr:PutRegistryScanningConfiguration ecr:PutReplicationConfiguration ecr:StartImageScan ecr:StartLifecyclePolicyPreview ecr:UpdatePullThroughCacheRule

服务前缀	操作
	ecr:UpdateRepositoryCreationTemplate ecr:UploadLayerPart ecr:ValidatePullThroughCacheRule

服务前缀	操作
ecr-public	ecr-public:BatchCheckLayerAvailability
	ecr-public:BatchDeleteImage
	ecr-public:CompleteLayerUpload
	ecr-public:CreateRepository
	ecr-public:DeleteRepository
	ecr-public:DeleteRepositoryPolicy
	ecr-public:DescribeImages
	ecr-public:DescribeRegistries
	ecr-public:DescribeRepositories
	ecr-public:GetAuthorizationToken
	ecr-public:GetRegistryCatalogData
	ecr-public:GetRepositoryCatalogData
	ecr-public:GetRepositoryPolicy
	ecr-public:InitiateLayerUpload
	ecr-public:PutImage
	ecr-public:PutRegistryCatalogData
	ecr-public:PutRepositoryCatalogData
	ecr-public:SetRepositoryPolicy
	ecr-public:UploadLayerPart

服务前缀	操作
ecs	ecs:CreateCapacityProvider ecs:CreateCluster ecs:CreateService ecs:CreateTaskSet ecs:DeleteAccountSetting ecs:DeleteAttributes ecs:DeleteCapacityProvider ecs:DeleteCluster ecs:DeleteExpressGatewayService ecs:DeleteService ecs:DeleteTaskDefinitions ecs:DeleteTaskSet ecs:DeregisterContainerInstance ecs:DeregisterTaskDefinition ecs:DescribeCapacityProviders ecs:DescribeClusters ecs:DescribeContainerInstances ecs:DescribeExpressGatewayService ecs:DescribeServiceDeployments ecs:DescribeServiceRevisions ecs:DescribeServices

服务前缀	操作
	ecs:DescribeTaskDefinition ecs:DescribeTaskSets ecs:DescribeTasks ecs:DiscoverPollEndpoint ecs:ExecuteCommand ecs:GetTaskProtection ecs:ListAccountSettings ecs:ListAttributes ecs:ListClusters ecs:ListContainerInstances ecs:ListServiceDeployments ecs:ListServices ecs:ListServicesByNamespace ecs:ListTaskDefinitionFamilies ecs:ListTaskDefinitions ecs:ListTasks ecs:PutAccountSetting ecs:PutAccountSettingDefault ecs:PutAttributes ecs:PutClusterCapacityProviders ecs:RegisterContainerInstance

服务前缀	操作
	ecs:RunTask ecs:StartTask ecs:StopServiceDeployment ecs:StopTask ecs:SubmitAttachmentStateChanges ecs:SubmitContainerStateChange ecs:SubmitTaskStateChange ecs:UpdateCapacityProvider ecs:UpdateCluster ecs:UpdateClusterSettings ecs:UpdateContainerAgent ecs:UpdateContainerInstancesState ecs:UpdateExpressGatewayService ecs:UpdateService ecs:UpdateServicePrimaryTaskSet ecs:UpdateTaskProtection ecs:UpdateTaskSet

服务前缀	操作
eks	eks:AssociateAccessPolicy
	eks:AssociateEncryptionConfig
	eks:AssociateIdentityProviderConfig
	eks:CreateAccessEntry
	eks:CreateAddon
	eks:CreateCluster
	eks:CreateEksAnywhereSubscription
	eks:CreateFargateProfile
	eks:CreateNodegroup
	eks:DeleteAccessEntry
	eks:DeleteAddon
	eks:DeleteCapability
	eks:DeleteCluster
	eks:DeleteEksAnywhereSubscription
	eks:DeleteFargateProfile
	eks:DeleteNodegroup
	eks:DeletePodIdentityAssociation
	eks:DeregisterCluster
	eks:DescribeAccessEntry
	eks:DescribeAddon
	eks:DescribeAddonConfiguration

服务前缀	操作
	eks:DescribeAddonVersions
	eks:DescribeCapability
	eks:DescribeCluster
	eks:DescribeClusterVersions
	eks:DescribeEksAnywhereSubscription
	eks:DescribeFargateProfile
	eks:DescribeIdentityProviderConfig
	eks:DescribeInsight
	eks:DescribeInsightsRefresh
	eks:DescribeNodegroup
	eks:DescribePodIdentityAssociation
	eks:DescribeUpdate
	eks:DisassociateAccessPolicy
	eks:DisassociateIdentityProviderConfig
	eks:ListAccessEntries
	eks:ListAccessPolicies
	eks:ListAddons
	eks:ListAssociatedAccessPolicies
	eks:ListCapabilities
	eks:ListClusters
	eks:ListEksAnywhereSubscriptions

服务前缀	操作
	eks:ListFargateProfiles
	eks:ListIdentityProviderConfigs
	eks:ListInsights
	eks:ListNodegroups
	eks:ListPodIdentityAssociations
	eks:ListUpdates
	eks:RegisterCluster
	eks:StartInsightsRefresh
	eks:UpdateAccessEntry
	eks:UpdateAddon
	eks:UpdateCapability
	eks:UpdateClusterConfig
	eks:UpdateClusterVersion
	eks:UpdateEksAnywhereSubscription
	eks:UpdateNodegroupConfig
	eks:UpdateNodegroupVersion
	eks:UpdatePodIdentityAssociation

服务前缀	操作
elasticache	elasticache:AuthorizeCacheSecurityGroupIngress elasticache:BatchApplyUpdateAction elasticache:BatchStopUpdateAction elasticache:CompleteMigration elasticache:CopyServerlessCacheSnapshot elasticache:CopySnapshot elasticache:CreateCacheCluster elasticache:CreateCacheParameterGroup elasticache:CreateCacheSecurityGroup elasticache:CreateCacheSubnetGroup elasticache:CreateGlobalReplicationGroup elasticache:CreateReplicationGroup elasticache:CreateServerlessCache elasticache:CreateServerlessCacheSnapshot elasticache:CreateSnapshot elasticache:CreateUser elasticache:CreateUserGroup elasticache:DecreaseNodeGroupsInGlobalReplicationGroup elasticache:DecreaseReplicaCount elasticache:DeleteCacheCluster elasticache:DeleteCacheParameterGroup

服务前缀	操作
	elasticache:DeleteCacheSecurityGroup elasticache:DeleteCacheSubnetGroup elasticache:DeleteGlobalReplicationGroup elasticache:DeleteReplicationGroup elasticache:DeleteServerlessCache elasticache:DeleteServerlessCacheSnapshot elasticache:DeleteSnapshot elasticache:DeleteUser elasticache:DeleteUserGroup elasticache:DescribeCacheClusters elasticache:DescribeCacheEngineVersions elasticache:DescribeCacheParameterGroups elasticache:DescribeCacheParameters elasticache:DescribeCacheSecurityGroups elasticache:DescribeCacheSubnetGroups elasticache:DescribeEngineDefaultParameters elasticache:DescribeEvents elasticache:DescribeGlobalReplicationGroups elasticache:DescribeReplicationGroups elasticache:DescribeReservedCacheNodes elasticache:DescribeReservedCacheNodesOfferings

服务前缀	操作
	elasticache:DescribeServerlessCacheSnapshots elasticache:DescribeServerlessCaches elasticache:DescribeServiceUpdates elasticache:DescribeSnapshots elasticache:DescribeUpdateActions elasticache:DescribeUserGroups elasticache:DescribeUsers elasticache:DisassociateGlobalReplicationGroup elasticache:ExportServerlessCacheSnapshot elasticache:FailoverGlobalReplicationGroup elasticache:IncreaseNodeGroupsInGlobalReplicationGroup elasticache:IncreaseReplicaCount elasticache:ListAllowedNodeTypeModifications elasticache:ModifyCacheCluster elasticache:ModifyCacheParameterGroup elasticache:ModifyCacheSubnetGroup elasticache:ModifyGlobalReplicationGroup elasticache:ModifyReplicationGroup elasticache:ModifyReplicationGroupShardConfiguration elasticache:ModifyServerlessCache elasticache:ModifyUser

服务前缀	操作
	elasticache:ModifyUserGroup elasticache:PurchaseReservedCacheNodesOffering elasticache:RebalanceSlotsInGlobalReplicationGroup elasticache:RebootCacheCluster elasticache:ResetCacheParameterGroup elasticache:RevokeCacheSecurityGroupIngress elasticache:StartMigration elasticache:TestFailover elasticache:TestMigration

服务前缀	操作
elasticbeanstalk	elasticbeanstalk:AbortEnvironmentUpdate
	elasticbeanstalk:ApplyEnvironmentManagedAction
	elasticbeanstalk:AssociateEnvironmentOperationsRole
	elasticbeanstalk:CheckDNSAvailability
	elasticbeanstalk:ComposeEnvironments
	elasticbeanstalk:CreateApplication
	elasticbeanstalk:CreateApplicationVersion
	elasticbeanstalk:CreateConfigurationTemplate
	elasticbeanstalk:CreateEnvironment
	elasticbeanstalk:CreatePlatformVersion
	elasticbeanstalk:CreateStorageLocation
	elasticbeanstalk:DeleteApplication
	elasticbeanstalk:DeleteApplicationVersion
	elasticbeanstalk:DeleteConfigurationTemplate
	elasticbeanstalk:DeleteEnvironmentConfiguration
	elasticbeanstalk:DeletePlatformVersion
	elasticbeanstalk:DescribeAccountAttributes
	elasticbeanstalk:DescribeApplicationVersions
	elasticbeanstalk:DescribeApplications
	elasticbeanstalk:DescribeConfigurationOptions
	elasticbeanstalk:DescribeConfigurationSettings

服务前缀	操作
	elasticbeanstalk:DescribeEnvironmentHealth
	elasticbeanstalk:DescribeEnvironmentManagedActionHistory
	elasticbeanstalk:DescribeEnvironmentManagedActions
	elasticbeanstalk:DescribeEnvironmentResources
	elasticbeanstalk:DescribeEnvironments
	elasticbeanstalk:DescribeEvents
	elasticbeanstalk:DescribeInstancesHealth
	elasticbeanstalk:DescribePlatformVersion
	elasticbeanstalk:DisassociateEnvironmentOperationsRole
	elasticbeanstalk:ListAvailableSolutionStacks
	elasticbeanstalk:ListPlatformBranches
	elasticbeanstalk:ListPlatformVersions
	elasticbeanstalk:RebuildEnvironment
	elasticbeanstalk:RequestEnvironmentInfo
	elasticbeanstalk:RestartAppServer
	elasticbeanstalk:RetrieveEnvironmentInfo
	elasticbeanstalk:SwapEnvironmentCNAMEs
	elasticbeanstalk:TerminateEnvironment
	elasticbeanstalk:UpdateApplication
	elasticbeanstalk:UpdateApplicationResourceLifecycle
	elasticbeanstalk:UpdateApplicationVersion

服务前缀	操作
	elasticbeanstalk:UpdateConfigurationTemplate elasticbeanstalk:UpdateEnvironment elasticbeanstalk:ValidateConfigurationSettings

服务前缀	操作
elasticfilesystem	elasticfilesystem:CreateAccessPoint elasticfilesystem:CreateFileSystem elasticfilesystem:CreateMountTarget elasticfilesystem:CreateReplicationConfiguration elasticfilesystem:DeleteAccessPoint elasticfilesystem:DeleteFileSystem elasticfilesystem:DeleteFileSystemPolicy elasticfilesystem:DeleteMountTarget elasticfilesystem:DeleteReplicationConfiguration elasticfilesystem:DescribeAccessPoints elasticfilesystem:DescribeAccountPreferences elasticfilesystem:DescribeBackupPolicy elasticfilesystem:DescribeFileSystemPolicy elasticfilesystem:DescribeFileSystems elasticfilesystem:DescribeLifecycleConfiguration elasticfilesystem:DescribeMountTargetSecurityGroups elasticfilesystem:DescribeMountTargets elasticfilesystem:DescribeReplicationConfigurations elasticfilesystem:ModifyMountTargetSecurityGroups elasticfilesystem:PutAccountPreferences elasticfilesystem:PutBackupPolicy

服务前缀	操作
	elasticfilesystem:PutFileSystemPolicy elasticfilesystem:PutLifecycleConfiguration elasticfilesystem:UpdateFileSystem elasticfilesystem:UpdateFileSystemProtection

服务前缀	操作
elasticloadbalancing	elasticloadbalancing:AddListenerCertificates
	elasticloadbalancing:AddTrustStoreRevocations
	elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
	elasticloadbalancing:AttachLoadBalancerToSubnets
	elasticloadbalancing:ConfigureHealthCheck
	elasticloadbalancing:CreateAppCookieStickinessPolicy
	elasticloadbalancing:CreateLBCookieStickinessPolicy
	elasticloadbalancing:CreateListener
	elasticloadbalancing:CreateLoadBalancer
	elasticloadbalancing:CreateLoadBalancerListeners
	elasticloadbalancing:CreateLoadBalancerPolicy
	elasticloadbalancing:CreateRule
	elasticloadbalancing:CreateTargetGroup
	elasticloadbalancing:CreateTrustStore
	elasticloadbalancing:DeleteListener
	elasticloadbalancing:DeleteLoadBalancer
	elasticloadbalancing:DeleteLoadBalancerListeners
	elasticloadbalancing:DeleteLoadBalancerPolicy
	elasticloadbalancing:DeleteRule
	elasticloadbalancing:DeleteSharedTrustStoreAssociation
	elasticloadbalancing:DeleteTargetGroup

服务前缀	操作
	elasticloadbalancing:DeleteTrustStore
	elasticloadbalancing:DeregisterInstancesFromLoadBalancer
	elasticloadbalancing:DeregisterTargets
	elasticloadbalancing:DescribeAccountLimits
	elasticloadbalancing:DescribeCapacityReservation
	elasticloadbalancing:DescribeInstanceHealth
	elasticloadbalancing:DescribeListenerAttributes
	elasticloadbalancing:DescribeListenerCertificates
	elasticloadbalancing:DescribeListeners
	elasticloadbalancing:DescribeLoadBalancerAttributes
	elasticloadbalancing:DescribeLoadBalancerPolicies
	elasticloadbalancing:DescribeLoadBalancerPolicyTypes
	elasticloadbalancing:DescribeLoadBalancers
	elasticloadbalancing:DescribeRules
	elasticloadbalancing:DescribeSSLPolicies
	elasticloadbalancing:DescribeTargetGroupAttributes
	elasticloadbalancing:DescribeTargetGroups
	elasticloadbalancing:DescribeTargetHealth
	elasticloadbalancing:DescribeTrustStoreAssociations
	elasticloadbalancing:DescribeTrustStoreRevocations
	elasticloadbalancing:DescribeTrustStores

服务前缀	操作
	elasticloadbalancing:DetachLoadBalancerFromSubnets
	elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer
	elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer
	elasticloadbalancing:GetResourcePolicy
	elasticloadbalancing:GetTrustStoreCaCertificatesBundle
	elasticloadbalancing:GetTrustStoreRevocationContent
	elasticloadbalancing:ModifyCapacityReservation
	elasticloadbalancing:ModifyIpPools
	elasticloadbalancing:ModifyListener
	elasticloadbalancing:ModifyLoadBalancerAttributes
	elasticloadbalancing:ModifyRule
	elasticloadbalancing:ModifyTargetGroup
	elasticloadbalancing:ModifyTargetGroupAttributes
	elasticloadbalancing:ModifyTrustStore
	elasticloadbalancing:RegisterInstancesWithLoadBalancer
	elasticloadbalancing:RegisterTargets
	elasticloadbalancing:RemoveListenerCertificates
	elasticloadbalancing:RemoveTrustStoreRevocations
	elasticloadbalancing:SetIpAddressType
	elasticloadbalancing:SetLoadBalancerListenerSSLCertificate
	elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer

服务前缀	操作
	elasticloadbalancing:SetLoadBalancerPoliciesOfListener elasticloadbalancing:SetRulePriorities elasticloadbalancing:SetSecurityGroups elasticloadbalancing:SetSubnets
elastictranscoder	elastictranscoder:CancelJob elastictranscoder:CreateJob elastictranscoder:CreatePipeline elastictranscoder:CreatePreset elastictranscoder:DeletePipeline elastictranscoder:DeletePreset elastictranscoder:ListJobsByPipeline elastictranscoder:ListJobsByStatus elastictranscoder:ListPipelines elastictranscoder:ListPresets elastictranscoder:ReadJob elastictranscoder:ReadPipeline elastictranscoder:ReadPreset elastictranscoder:TestRole elastictranscoder:UpdatePipeline elastictranscoder:UpdatePipelineNotifications elastictranscoder:UpdatePipelineStatus

服务前缀	操作
emr-containers	emr-containers:CancelJobRun
	emr-containers:CreateJobTemplate
	emr-containers:CreateManagedEndpoint
	emr-containers:CreateSecurityConfiguration
	emr-containers:CreateVirtualCluster
	emr-containers:DeleteJobTemplate
	emr-containers:DeleteManagedEndpoint
	emr-containers:DeleteVirtualCluster
	emr-containers:DescribeJobRun
	emr-containers:DescribeJobTemplate
	emr-containers:DescribeManagedEndpoint
	emr-containers:DescribeSecurityConfiguration
	emr-containers:DescribeVirtualCluster
	emr-containers:GetManagedEndpointSessionCredentials
	emr-containers:ListJobRuns
	emr-containers:ListJobTemplates
	emr-containers:ListManagedEndpoints
	emr-containers:ListSecurityConfigurations
	emr-containers:ListVirtualClusters
	emr-containers:StartJobRun

服务前缀	操作
emr-serverless	emr-serverless:CancelJobRun emr-serverless:CreateApplication emr-serverless:DeleteApplication emr-serverless:GetApplication emr-serverless:GetDashboardForJobRun emr-serverless:GetJobRun emr-serverless:ListApplications emr-serverless:ListJobRunAttempts emr-serverless:ListJobRuns emr-serverless:StartApplication emr-serverless:StartJobRun emr-serverless:StopApplication emr-serverless:UpdateApplication

服务前缀	操作
es	es:AcceptInboundConnection
	es:AcceptInboundCrossClusterSearchConnection
	es:AssociatePackage
	es:AuthorizeVpcEndpointAccess
	es:CancelElasticsearchServiceSoftwareUpdate
	es:CancelServiceSoftwareUpdate
	es:CreateDomain
	es:CreateElasticsearchDomain
	es:CreateIndex
	es:CreateOutboundConnection
	es:CreateOutboundCrossClusterSearchConnection
	es:CreatePackage
	es:CreateVpcEndpoint
	es:DeleteDomain
	es:DeleteElasticsearchDomain
	es:DeleteElasticsearchServiceRole
	es:DeleteInboundConnection
	es:DeleteInboundCrossClusterSearchConnection
	es:DeleteIndex
	es:DeleteOutboundConnection
	es:DeleteOutboundCrossClusterSearchConnection

服务前缀	操作
	es:DeletePackage
	es:DeleteVpcEndpoint
	es:DescribeDomain
	es:DescribeDomainAutoTunes
	es:DescribeDomainChangeProgress
	es:DescribeDomainConfig
	es:DescribeDomainHealth
	es:DescribeDomainNodes
	es:DescribeDomains
	es:DescribeDryRunProgress
	es:DescribeElasticsearchDomain
	es:DescribeElasticsearchDomainConfig
	es:DescribeElasticsearchDomains
	es:DescribeElasticsearchInstanceTypeLimits
	es:DescribeInboundConnections
	es:DescribeInboundCrossClusterSearchConnections
	es:DescribeInstanceTypeLimits
	es:DescribeOutboundConnections
	es:DescribeOutboundCrossClusterSearchConnections
	es:DescribePackages
	es:DescribeReservedElasticsearchInstanceOfferings

服务前缀	操作
	es:DescribeReservedElasticsearchInstances
	es:DescribeReservedInstanceOfferings
	es:DescribeReservedInstances
	es:DescribeVpcEndpoints
	es:DissociatePackage
	es:DissociatePackages
	es:GetCompatibleElasticsearchVersions
	es:GetCompatibleVersions
	es:GetDataSource
	es:GetDomainMaintenanceStatus
	es:GetPackageVersionHistory
	es:GetUpgradeHistory
	es:GetUpgradeStatus
	es:ListDataSources
	es:ListDomainNames
	es:ListDomainsForPackage
	es:ListElasticsearchInstanceTypes
	es:ListElasticsearchVersions
	es:ListInstanceTypeDetails
	es:ListPackagesForDomain
	es:ListScheduledActions

服务前缀	操作
	es:ListVersions
	es:ListVpcEndpointAccess
	es:ListVpcEndpoints
	es:ListVpcEndpointsForDomain
	es:PurchaseReservedElasticsearchInstanceOffering
	es:PurchaseReservedInstanceOffering
	es:RejectInboundConnection
	es:RejectInboundCrossClusterSearchConnection
	es:RevokeVpcEndpointAccess
	es:StartDomainMaintenance
	es:StartElasticsearchServiceSoftwareUpdate
	es:StartServiceSoftwareUpdate
	es:UpdateDataSource
	es:UpdateDomainConfig
	es:UpdateElasticsearchDomainConfig
	es:UpdateIndex
	es:UpdatePackage
	es:UpdatePackageScope
	es:UpdateScheduledAction
	es:UpdateVpcEndpoint
	es:UpgradeDomain

服务前缀	操作
	es:UpgradeElasticsearchDomain

服务前缀	操作
events	events:ActivateEventSource events:CancelReplay events:CreateApiDestination events:CreateArchive events:CreateConnection events:CreateEndpoint events:CreateEventBus events:CreatePartnerEventSource events:DeactivateEventSource events:DeauthorizeConnection events:DeleteApiDestination events:DeleteArchive events:DeleteConnection events:DeleteEndpoint events:DeleteEventBus events:DeletePartnerEventSource events:DeleteRule events:DescribeApiDestination events:DescribeArchive events:DescribeConnection events:DescribeEndpoint

服务前缀	操作
	events:DescribeEventBus events:DescribeEventSource events:DescribePartnerEventSource events:DescribeReplay events:DescribeRule events:DisableRule events:EnableRule events:ListApiDestinations events:ListArchives events:ListConnections events:ListEndpoints events:ListEventBuses events:ListEventSources events:ListPartnerEventSourceAccounts events:ListPartnerEventSources events:ListReplays events:ListRuleNamesByTarget events:ListRules events:ListTargetsByRule events:PutPermission events:PutRule

服务前缀	操作
	events:PutTargets events:RemovePermission events:RemoveTargets events:StartReplay events:TestEventPattern events:UpdateApiDestination events:UpdateArchive events:UpdateConnection events:UpdateEndpoint events:UpdateEventBus

服务前缀	操作
evidently	evidently:CreateExperiment
	evidently:CreateFeature
	evidently:CreateLaunch
	evidently:CreateProject
	evidently:CreateSegment
	evidently:DeleteExperiment
	evidently:DeleteFeature
	evidently:DeleteLaunch
	evidently:DeleteProject
	evidently:DeleteSegment
	evidently:GetExperiment
	evidently:GetExperimentResults
	evidently:GetFeature
	evidently:GetLaunch
	evidently:GetProject
	evidently:GetSegment
	evidently:ListExperiments
	evidently:ListFeatures
	evidently:ListLaunches
	evidently:ListProjects
	evidently:ListSegmentReferences

服务前缀	操作
	evidently:ListSegments
	evidently:StartExperiment
	evidently:StartLaunch
	evidently:StopExperiment
	evidently:StopLaunch
	evidently:TestSegmentPattern
	evidently:UpdateExperiment
	evidently:UpdateFeature
	evidently:UpdateLaunch
	evidently:UpdateProject
	evidently:UpdateProjectDataDelivery

服务前缀	操作
finspace	finspace:CreateEnvironment finspace:CreateKxChangeset finspace:CreateKxCluster finspace:CreateKxDatabase finspace:CreateKxDataview finspace:CreateKxEnvironment finspace:CreateKxScalingGroup finspace:CreateKxUser finspace:CreateKxVolume finspace:CreateUser finspace:DeleteEnvironment finspace:DeleteKxCluster finspace:DeleteKxClusterNode finspace:DeleteKxDatabase finspace:DeleteKxDataview finspace:DeleteKxEnvironment finspace:DeleteKxScalingGroup finspace:DeleteKxUser finspace:DeleteKxVolume finspace:GetEnvironment finspace:GetKxChangeset

服务前缀	操作
	finspace:GetKxCluster finspace:GetKxConnectionString finspace:GetKxDatabase finspace:GetKxDataview finspace:GetKxEnvironment finspace:GetKxScalingGroup finspace:GetKxUser finspace:GetKxVolume finspace:GetLoadSampleDataSetGroupIntoEnvironmentStatus finspace:GetUser finspace:ListEnvironments finspace:ListKxChangesets finspace:ListKxClusterNodes finspace:ListKxClusters finspace:ListKxDatabases finspace:ListKxDataviews finspace:ListKxEnvironments finspace:ListKxScalingGroups finspace:ListKxUsers finspace:ListKxVolumes finspace:ListUsers

服务前缀	操作
	finspace:LoadSampleDataSetGroupIntoEnvironment finspace:ResetUserPassword finspace:UpdateEnvironment finspace:UpdateKxClusterCodeConfiguration finspace:UpdateKxClusterDatabases finspace:UpdateKxDatabase finspace:UpdateKxDataview finspace:UpdateKxEnvironment finspace:UpdateKxEnvironmentNetwork finspace:UpdateKxUser finspace:UpdateKxVolume finspace:UpdateUser
Firehose	firehose:CreateDeliveryStream firehose:DeleteDeliveryStream firehose:DescribeDeliveryStream firehose:ListDeliveryStreams firehose:StartDeliveryStreamEncryption firehose:StopDeliveryStreamEncryption firehose:UpdateDestination

服务前缀	操作
fis	fis:CreateExperimentTemplate fis:CreateTargetAccountConfiguration fis:DeleteExperimentTemplate fis:DeleteTargetAccountConfiguration fis:GetAction fis:GetExperiment fis:GetExperimentTargetAccountConfiguration fis:GetExperimentTemplate fis:GetSafetyLever fis:GetTargetAccountConfiguration fis:GetTargetResourceType fis:ListActions fis:ListExperimentResolvedTargets fis:ListExperimentTargetAccountConfigurations fis:ListExperimentTemplates fis:ListExperiments fis:ListTargetAccountConfigurations fis:ListTargetResourceTypes fis:StartExperiment fis:StopExperiment fis:UpdateExperimentTemplate

服务前缀	操作
	fis:UpdateSafetyLeverState fis:UpdateTargetAccountConfiguration

服务前缀	操作
fms	fms:AssociateAdminAccount
	fms:AssociateThirdPartyFirewall
	fms:BatchAssociateResource
	fms:BatchDisassociateResource
	fms:DeleteAppsList
	fms:DeleteNotificationChannel
	fms:DeletePolicy
	fms:DeleteProtocolsList
	fms:DeleteResourceSet
	fms:DisassociateAdminAccount
	fms:DisassociateThirdPartyFirewall
	fms:GetAdminAccount
	fms:GetAdminScope
	fms:GetAppsList
	fms:GetComplianceDetail
	fms:GetNotificationChannel
	fms:GetPolicy
	fms:GetProtectionStatus
	fms:GetProtocolsList
	fms:GetResourceSet
	fms:GetThirdPartyFirewallAssociationStatus

服务前缀	操作
	fms:GetViolationDetails
	fms:ListAdminAccountsForOrganization
	fms:ListAdminsManagingAccount
	fms:ListAppsLists
	fms:ListComplianceStatus
	fms:ListDiscoveredResources
	fms:ListMemberAccounts
	fms:ListPolicies
	fms:ListProtocolsLists
	fms:ListResourceSetResources
	fms:ListResourceSets
	fms:ListThirdPartyFirewallFirewallPolicies
	fms:PutAdminAccount
	fms:PutAppsList
	fms:PutNotificationChannel
	fms:PutPolicy
	fms:PutProtocolsList
	fms:PutResourceSet

服务前缀	操作
frauddetector	frauddetector:BatchCreateVariable frauddetector:BatchGetVariable frauddetector:CancelBatchImportJob frauddetector:CancelBatchPredictionJob frauddetector:CreateBatchImportJob frauddetector:CreateBatchPredictionJob frauddetector:CreateDetectorVersion frauddetector:CreateList frauddetector:CreateModel frauddetector:CreateModelVersion frauddetector:CreateRule frauddetector:CreateVariable frauddetector:DeleteBatchImportJob frauddetector:DeleteBatchPredictionJob frauddetector:DeleteDetector frauddetector:DeleteDetectorVersion frauddetector:DeleteEntityType frauddetector:DeleteEvent frauddetector:DeleteEventType frauddetector:DeleteEventsByEventType frauddetector:DeleteExternalModel

服务前缀	操作
	frauddetector:DeleteLabel frauddetector:DeleteList frauddetector:DeleteModel frauddetector:DeleteModelVersion frauddetector:DeleteOutcome frauddetector:DeleteRule frauddetector:DeleteVariable frauddetector:DescribeDetector frauddetector:DescribeModelVersions frauddetector:GetBatchImportJobs frauddetector:GetBatchPredictionJobs frauddetector:GetDeleteEventsByEventTypeStatus frauddetector:GetDetectorVersion frauddetector:GetDetectors frauddetector:GetEntityTypes frauddetector:GetEvent frauddetector:GetEventPrediction frauddetector:GetEventPredictionMetadata frauddetector:GetEventTypes frauddetector:GetExternalModels frauddetector:GetKMSEncryptionKey

服务前缀	操作
	frauddetector:GetLabels frauddetector:GetListElements frauddetector:GetListsMetadata frauddetector:GetModelVersion frauddetector:GetModels frauddetector:GetOutcomes frauddetector:GetRules frauddetector:GetVariables frauddetector:ListEventPredictions frauddetector:PutDetector frauddetector:PutEntityType frauddetector:PutEventType frauddetector:PutExternalModel frauddetector:PutKMSEncryptionKey frauddetector:PutLabel frauddetector:PutOutcome frauddetector:SendEvent frauddetector:UpdateDetectorVersion frauddetector:UpdateDetectorVersionMetadata frauddetector:UpdateDetectorVersionStatus frauddetector:UpdateEventLabel

服务前缀	操作
	frauddetector:UpdateList frauddetector:UpdateModel frauddetector:UpdateModelVersion frauddetector:UpdateModelVersionStatus frauddetector:UpdateRuleMetadata frauddetector:UpdateRuleVersion frauddetector:UpdateVariable

服务前缀	操作
fsx	fsx:AssociateFileSystemAliases
	fsx:CancelDataRepositoryTask
	fsx:CopyBackup
	fsx:CreateDataRepositoryTask
	fsx:CreateFileCache
	fsx:CreateFileSystem
	fsx:CreateFileSystemFromBackup
	fsx:CreateSnapshot
	fsx:CreateStorageVirtualMachine
	fsx:CreateVolume
	fsx:CreateVolumeFromBackup
	fsx:DeleteBackup
	fsx:DeleteFileCache
	fsx:DeleteFileSystem
	fsx:DeleteSnapshot
	fsx:DeleteStorageVirtualMachine
	fsx:DeleteVolume
	fsx:DescribeBackups
	fsx:DescribeDataRepositoryAssociations
	fsx:DescribeDataRepositoryTasks
	fsx:DescribeFileCaches

服务前缀	操作
	fsx:DescribeFileSystemAliases
	fsx:DescribeFileSystems
	fsx:DescribeS3AccessPointAttachments
	fsx:DescribeSharedVpcConfiguration
	fsx:DescribeSnapshots
	fsx:DescribeStorageVirtualMachines
	fsx:DescribeVolumes
	fsx:DetachAndDeleteS3AccessPoint
	fsx:DisassociateFileSystemAliases
	fsx:ReleaseFileSystemNfsV3Locks
	fsx:RestoreVolumeFromSnapshot
	fsx:StartMisconfiguredStateRecovery
	fsx:UpdateDataRepositoryAssociation
	fsx:UpdateFileCache
	fsx:UpdateFileSystem
	fsx:UpdateSharedVpcConfiguration
	fsx:UpdateSnapshot
	fsx:UpdateStorageVirtualMachine
	fsx:UpdateVolume

服务前缀	操作
gamelift	gamelift:AcceptMatch gamelift:ClaimGameServer gamelift:CreateAlias gamelift:CreateBuild gamelift:CreateContainerGroupDefinition gamelift:CreateFleet gamelift:CreateFleetLocations gamelift:CreateGameServerGroup gamelift:CreateGameSession gamelift:CreateGameSessionQueue gamelift:CreateLocation gamelift:CreateMatchmakingConfiguration gamelift:CreateMatchmakingRuleSet gamelift:CreatePlayerSession gamelift:CreatePlayerSessions gamelift:CreateScript gamelift:CreateVpcPeeringAuthorization gamelift:CreateVpcPeeringConnection gamelift:DeleteAlias gamelift:DeleteBuild gamelift:DeleteContainerGroupDefinition

服务前缀	操作
	gamelift:DeleteFleet gamelift:DeleteFleetLocations gamelift:DeleteGameServerGroup gamelift:DeleteGameSessionQueue gamelift:DeleteLocation gamelift:DeleteMatchmakingConfiguration gamelift:DeleteMatchmakingRuleSet gamelift:DeleteScalingPolicy gamelift:DeleteScript gamelift:DeleteVpcPeeringAuthorization gamelift:DeleteVpcPeeringConnection gamelift:DeregisterCompute gamelift:DeregisterGameServer gamelift:DescribeAlias gamelift:DescribeBuild gamelift:DescribeCompute gamelift:DescribeContainerFleet gamelift:DescribeContainerGroupDefinition gamelift:DescribeEC2InstanceLimits gamelift:DescribeFleetAttributes gamelift:DescribeFleetCapacity

服务前缀	操作
	gamelift:DescribeFleetEvents gamelift:DescribeFleetLocationAttributes gamelift:DescribeFleetLocationCapacity gamelift:DescribeFleetLocationUtilization gamelift:DescribeFleetPortSettings gamelift:DescribeFleetUtilization gamelift:DescribeGameServer gamelift:DescribeGameServerGroup gamelift:DescribeGameServerInstances gamelift:DescribeGameSessionDetails gamelift:DescribeGameSessionPlacement gamelift:DescribeGameSessionQueues gamelift:DescribeGameSessions gamelift:DescribeInstances gamelift:DescribeMatchmaking gamelift:DescribeMatchmakingConfigurations gamelift:DescribeMatchmakingRuleSets gamelift:DescribePlayerSessions gamelift:DescribeRuntimeConfiguration gamelift:DescribeScalingPolicies gamelift:DescribeScript

服务前缀	操作
	gamelift:DescribeVpcPeeringAuthorizations gamelift:DescribeVpcPeeringConnections gamelift:GetComputeAccess gamelift:GetComputeAuthToken gamelift:GetGameSessionLogUrl gamelift:GetInstanceAccess gamelift:ListAliases gamelift:ListBuilds gamelift:ListCompute gamelift:ListContainerFleets gamelift:ListContainerGroupDefinitionVersions gamelift:ListContainerGroupDefinitions gamelift:ListFleetDeployments gamelift:ListFleets gamelift:ListGameServerGroups gamelift:ListGameServers gamelift:ListLocations gamelift:ListScripts gamelift:PutScalingPolicy gamelift:RegisterCompute gamelift:RegisterGameServer

服务前缀	操作
	gamelift:RequestUploadCredentials gamelift:ResolveAlias gamelift:ResumeGameServerGroup gamelift:SearchGameSessions gamelift:StartFleetActions gamelift:StartGameSessionPlacement gamelift:StartMatchBackfill gamelift:StartMatchmaking gamelift:StopFleetActions gamelift:StopGameSessionPlacement gamelift:StopMatchmaking gamelift:SuspendGameServerGroup gamelift:TerminateGameSession gamelift:UpdateAlias gamelift:UpdateBuild gamelift:UpdateContainerGroupDefinition gamelift:UpdateFleetAttributes gamelift:UpdateFleetCapacity gamelift:UpdateFleetPortSettings gamelift:UpdateGameServer gamelift:UpdateGameServerGroup

服务前缀	操作
	gamelift:UpdateGameSession gamelift:UpdateGameSessionQueue gamelift:UpdateMatchmakingConfiguration gamelift:UpdateRuntimeConfiguration gamelift:UpdateScript gamelift:ValidateMatchmakingRuleSet

服务前缀	操作
geo	geo:AssociateTrackerConsumer
	geo:BatchDeleteDevicePositionHistory
	geo:BatchDeleteGeofence
	geo:BatchEvaluateGeofences
	geo:BatchGetDevicePosition
	geo:BatchPutGeofence
	geo:BatchUpdateDevicePosition
	geo:CalculateRoute
	geo:CalculateRouteMatrix
	geo:CreateGeofenceCollection
	geo:CreateMap
	geo:CreatePlaceIndex
	geo:CreateRouteCalculator
	geo:CreateTracker
	geo:DeleteGeofenceCollection
	geo:DeleteKey
	geo:DeleteMap
	geo:DeletePlaceIndex
	geo:DeleteRouteCalculator
	geo:DeleteTracker
	geo:DescribeGeofenceCollection

服务前缀	操作
	geo:DescribeKey
	geo:DescribeMap
	geo:DescribePlaceIndex
	geo:DescribeRouteCalculator
	geo:DescribeTracker
	geo:DisassociateTrackerConsumer
	geo:ForecastGeofenceEvents
	geo:GetDevicePosition
	geo:GetDevicePositionHistory
	geo:GetGeofence
	geo:GetMapGlyphs
	geo:GetMapSprites
	geo:GetMapStyleDescriptor
	geo:GetMapTile
	geo:GetPlace
	geo:ListDevicePositions
	geo:ListGeofenceCollections
	geo:ListGeofences
	geo:ListKeys
	geo:ListMaps
	geo:ListPlaceIndexes

服务前缀	操作
	geo:ListRouteCalculators
	geo:ListTrackerConsumers
	geo:ListTrackers
	geo:PutGeofence
	geo:SearchPlaceIndexForPosition
	geo:SearchPlaceIndexForSuggestions
	geo:SearchPlaceIndexForText
	geo:UpdateGeofenceCollection
	geo:UpdateKey
	geo:UpdateMap
	geo:UpdatePlaceIndex
	geo:UpdateRouteCalculator
	geo:UpdateTracker
	geo:VerifyDevicePosition

服务前缀	操作
glacier	glacier:AbortMultipartUpload glacier:AbortVaultLock glacier:CompleteMultipartUpload glacier:CompleteVaultLock glacier:CreateVault glacier:DeleteArchive glacier:DeleteVault glacier:DeleteVaultAccessPolicy glacier:DeleteVaultNotifications glacier:DescribeJob glacier:DescribeVault glacier:GetDataRetrievalPolicy glacier:GetJobOutput glacier:GetVaultAccessPolicy glacier:GetVaultLock glacier:GetVaultNotifications glacier:InitiateJob glacier:InitiateMultipartUpload glacier:InitiateVaultLock glacier:ListJobs glacier:ListMultipartUploads

服务前缀	操作
	glacier:ListParts glacier:ListProvisionedCapacity glacier:ListVaults glacier:PurchaseProvisionedCapacity glacier:SetDataRetrievalPolicy glacier:SetVaultAccessPolicy glacier:SetVaultNotifications glacier:UploadArchive glacier:UploadMultipartPart

服务前缀	操作
grafana	grafana:AssociateLicense
	grafana:CreateWorkspace
	grafana:CreateWorkspaceApiKey
	grafana:CreateWorkspaceServiceAccount
	grafana:CreateWorkspaceServiceAccountToken
	grafana:DeleteWorkspace
	grafana:DeleteWorkspaceApiKey
	grafana:DeleteWorkspaceServiceAccount
	grafana:DeleteWorkspaceServiceAccountToken
	grafana:DescribeWorkspace
	grafana:DescribeWorkspaceAuthentication
	grafana:DescribeWorkspaceConfiguration
	grafana:DisassociateLicense
	grafana:ListPermissions
	grafana:ListVersions
	grafana:ListWorkspaceServiceAccountTokens
	grafana:ListWorkspaceServiceAccounts
	grafana:ListWorkspaces
	grafana:UpdatePermissions
	grafana:UpdateWorkspace
	grafana:UpdateWorkspaceAuthentication

服务前缀	操作
	grafana:UpdateWorkspaceConfiguration

服务前缀	操作
greengrass	greengrass:AssociateRoleToGroup greengrass:AssociateServiceRoleToAccount greengrass:BatchAssociateClientDeviceWithCoreDevice greengrass:BatchDisassociateClientDeviceFromCoreDevice greengrass:CancelDeployment greengrass:CreateComponentVersion greengrass:CreateConnectorDefinition greengrass:CreateConnectorDefinitionVersion greengrass:CreateCoreDefinition greengrass:CreateCoreDefinitionVersion greengrass:CreateDeployment greengrass:CreateDeviceDefinition greengrass:CreateDeviceDefinitionVersion greengrass:CreateFunctionDefinition greengrass:CreateFunctionDefinitionVersion greengrass:CreateGroup greengrass:CreateGroupCertificateAuthority greengrass:CreateGroupVersion greengrass:CreateLoggerDefinition greengrass:CreateLoggerDefinitionVersion greengrass:CreateResourceDefinition

服务前缀	操作
	greengrass:CreateResourceDefinitionVersion greengrass:CreateSoftwareUpdateJob greengrass:CreateSubscriptionDefinition greengrass:CreateSubscriptionDefinitionVersion greengrass:DeleteComponent greengrass:DeleteConnectorDefinition greengrass:DeleteCoreDefinition greengrass:DeleteCoreDevice greengrass:DeleteDeployment greengrass:DeleteDeviceDefinition greengrass:DeleteFunctionDefinition greengrass:DeleteGroup greengrass:DeleteLoggerDefinition greengrass:DeleteResourceDefinition greengrass:DeleteSubscriptionDefinition greengrass:DescribeComponent greengrass:DisassociateRoleFromGroup greengrass:DisassociateServiceRoleFromAccount greengrass:GetAssociatedRole greengrass:GetBulkDeploymentStatus greengrass:GetComponent

服务前缀	操作
	greengrass:GetComponentVersionArtifact greengrass:GetConnectivityInfo greengrass:GetConnectorDefinition greengrass:GetConnectorDefinitionVersion greengrass:GetCoreDefinition greengrass:GetCoreDefinitionVersion greengrass:GetCoreDevice greengrass:GetDeployment greengrass:GetDeploymentStatus greengrass:GetDeviceDefinition greengrass:GetDeviceDefinitionVersion greengrass:GetFunctionDefinition greengrass:GetFunctionDefinitionVersion greengrass:GetGroup greengrass:GetGroupCertificateAuthority greengrass:GetGroupCertificateConfiguration greengrass:GetGroupVersion greengrass:GetLoggerDefinition greengrass:GetLoggerDefinitionVersion greengrass:GetResourceDefinition greengrass:GetResourceDefinitionVersion

服务前缀	操作
	greengrass:GetServiceRoleForAccount greengrass:GetSubscriptionDefinition greengrass:GetSubscriptionDefinitionVersion greengrass:GetThingRuntimeConfiguration greengrass:ListBulkDeploymentDetailedReports greengrass:ListBulkDeployments greengrass:ListClientDevicesAssociatedWithCoreDevice greengrass:ListComponentVersions greengrass:ListComponents greengrass:ListConnectorDefinitionVersions greengrass:ListConnectorDefinitions greengrass:ListCoreDefinitionVersions greengrass:ListCoreDefinitions greengrass:ListCoreDevices greengrass:ListDeployments greengrass:ListDeviceDefinitionVersions greengrass:ListDeviceDefinitions greengrass:ListEffectiveDeployments greengrass:ListFunctionDefinitionVersions greengrass:ListFunctionDefinitions greengrass:ListGroupCertificateAuthorities

服务前缀	操作
	greengrass:ListGroupVersions greengrass:ListGroups greengrass:ListInstalledComponents greengrass:ListLoggerDefinitionVersions greengrass:ListLoggerDefinitions greengrass:ListResourceDefinitionVersions greengrass:ListResourceDefinitions greengrass:ListSubscriptionDefinitionVersions greengrass:ListSubscriptionDefinitions greengrass:ResetDeployments greengrass:StartBulkDeployment greengrass:StopBulkDeployment greengrass:UpdateConnectivityInfo greengrass:UpdateConnectorDefinition greengrass:UpdateCoreDefinition greengrass:UpdateDeviceDefinition greengrass:UpdateFunctionDefinition greengrass:UpdateGroup greengrass:UpdateGroupCertificateConfiguration greengrass:UpdateLoggerDefinition greengrass:UpdateResourceDefinition

服务前缀	操作
	greengrass:UpdateSubscriptionDefinition greengrass:UpdateThingRuntimeConfiguration

服务前缀	操作
groundstation	groundstation:CancelContact
	groundstation:CreateConfig
	groundstation:CreateDataflowEndpointGroup
	groundstation:CreateDataflowEndpointGroupV2
	groundstation:CreateEphemeris
	groundstation:CreateMissionProfile
	groundstation:DeleteConfig
	groundstation:DeleteDataflowEndpointGroup
	groundstation:DeleteEphemeris
	groundstation:DeleteMissionProfile
	groundstation:DescribeContact
	groundstation:DescribeEphemeris
	groundstation:GetConfig
	groundstation:GetDataflowEndpointGroup
	groundstation:GetMinuteUsage
	groundstation:GetMissionProfile
	groundstation:GetSatellite
	groundstation:ListConfigs
	groundstation:ListContacts
	groundstation:ListDataflowEndpointGroups
	groundstation:ListEphemerides

服务前缀	操作
	groundstation:ListGroundStations
	groundstation:ListMissionProfiles
	groundstation:ListSatellites
	groundstation:RegisterAgent
	groundstation:ReserveContact
	groundstation:UpdateAgentStatus
	groundstation:UpdateConfig
	groundstation:UpdateEphemeris
	groundstation:UpdateMissionProfile

服务前缀	操作
guardduty	guardduty:AcceptAdministratorInvitation guardduty:AcceptInvitation guardduty:ArchiveFindings guardduty:CreateDetector guardduty:CreateFilter guardduty:CreateIPSet guardduty:CreateMalwareProtectionPlan guardduty:CreateMembers guardduty:CreatePublishingDestination guardduty:CreateSampleFindings guardduty:CreateThreatEntitySet guardduty:CreateThreatIntelSet guardduty:CreateTrustedEntitySet guardduty:DeclineInvitations guardduty:DeleteDetector guardduty:DeleteFilter guardduty:DeleteIPSet guardduty:DeleteInvitations guardduty:DeleteMalwareProtectionPlan guardduty:DeleteMembers guardduty:DeletePublishingDestination

服务前缀	操作
	guardduty:DeleteThreatEntitySet guardduty:DeleteThreatIntelSet guardduty:DeleteTrustedEntitySet guardduty:DescribeMalwareScans guardduty:DescribeOrganizationConfiguration guardduty:DescribePublishingDestination guardduty:DisableOrganizationAdminAccount guardduty:DisassociateFromAdministratorAccount guardduty:DisassociateFromMasterAccount guardduty:DisassociateMembers guardduty:EnableOrganizationAdminAccount guardduty:GetAdministratorAccount guardduty:GetCoverageStatistics guardduty:GetDetector guardduty:GetFilter guardduty:GetFindings guardduty:GetFindingsStatistics guardduty:GetIPSet guardduty:GetInvitationsCount guardduty:GetMalwareProtectionPlan guardduty:GetMalwareScan

服务前缀	操作
	guardduty:GetMalwareScanSettings guardduty:GetMasterAccount guardduty:GetMemberDetectors guardduty:GetMembers guardduty:GetOrganizationStatistics guardduty:GetRemainingFreeTrialDays guardduty:GetThreatEntitySet guardduty:GetThreatIntelSet guardduty:GetTrustedEntitySet guardduty:GetUsageStatistics guardduty:InviteMembers guardduty:ListCoverage guardduty:ListDetectors guardduty:ListFilters guardduty:ListFindings guardduty:ListIPSets guardduty:ListInvitations guardduty:ListMalwareProtectionPlans guardduty:ListMalwareScans guardduty:ListMembers guardduty:ListOrganizationAdminAccounts

服务前缀	操作
	guardduty:ListPublishingDestinations guardduty:ListThreatEntitySets guardduty:ListThreatIntelSets guardduty:ListTrustedEntitySets guardduty:StartMalwareScan guardduty:StartMonitoringMembers guardduty:StopMonitoringMembers guardduty:UnarchiveFindings guardduty:UpdateDetector guardduty:UpdateFilter guardduty:UpdateFindingsFeedback guardduty:UpdateIPSet guardduty:UpdateMalwareProtectionPlan guardduty:UpdateMalwareScanSettings guardduty:UpdateMemberDetectors guardduty:UpdateOrganizationConfiguration guardduty:UpdatePublishingDestination guardduty:UpdateThreatEntitySet guardduty:UpdateThreatIntelSet guardduty:UpdateTrustedEntitySet

服务前缀	操作
healthlake	healthlake:CancelFHIRExportJobWithDelete
	healthlake:CreateFHIRDatastore
	healthlake:CreateResource
	healthlake:DeleteFHIRDatastore
	healthlake:DeleteResource
	healthlake:DescribeFHIRDatastore
	healthlake:DescribeFHIRExportJob
	healthlake:DescribeFHIRExportJobWithGet
	healthlake:DescribeFHIRImportJob
	healthlake:GetCapabilities
	healthlake:ListFHIRDatastores
	healthlake:ListFHIRExportJobs
	healthlake:ListFHIRImportJobs
	healthlake:ReadResource
	healthlake:SearchEverything
	healthlake:SearchWithGet
	healthlake:SearchWithPost
	healthlake:StartFHIRExportJob
	healthlake:StartFHIRExportJobWithPost
	healthlake:StartFHIRImportJob
	healthlake:UpdateResource

服务前缀	操作
honeycode	honeycode:BatchCreateTableRows honeycode:BatchDeleteTableRows honeycode:BatchUpdateTableRows honeycode:BatchUpsertTableRows honeycode:DescribeTableDataImportJob honeycode:GetScreenData honeycode:InvokeScreenAutomation honeycode:ListTableColumns honeycode:ListTableRows honeycode:ListTables honeycode:QueryTableRows honeycode:StartTableDataImportJob

服务前缀	操作
IAM	iam:AddClientIDToOpenIDConnectProvider
	iam:AddRoleToInstanceProfile
	iam:AddUserToGroup
	iam:AttachGroupPolicy
	iam:AttachRolePolicy
	iam:AttachUserPolicy
	iam:ChangePassword
	iam:CreateAccessKey
	iam:CreateAccountAlias
	iam:CreateGroup
	iam:CreateInstanceProfile
	iam:CreateLoginProfile
	iam:CreateOpenIDConnectProvider
	iam:CreatePolicy
	iam:CreatePolicyVersion
	iam:CreateRole
	iam:CreateSAMLProvider
	iam:CreateServiceLinkedRole
	iam:CreateServiceSpecificCredential
	iam:CreateUser
	iam:CreateVirtualMFADevice

服务前缀	操作
	iam:DeactivateMFADevice
	iam:DeleteAccessKey
	iam:DeleteAccountAlias
	iam:DeleteAccountPasswordPolicy
	iam:DeleteCloudFrontPublicKey
	iam:DeleteGroup
	iam:DeleteGroupPolicy
	iam:DeleteInstanceProfile
	iam:DeleteLoginProfile
	iam:DeleteOpenIDConnectProvider
	iam:DeletePolicy
	iam:DeletePolicyVersion
	iam:DeleteRole
	iam:DeleteRolePermissionsBoundary
	iam:DeleteRolePolicy
	iam:DeleteSAMLProvider
	iam:DeleteSSHPublicKey
	iam:DeleteServerCertificate
	iam:DeleteServiceLinkedRole
	iam:DeleteServiceSpecificCredential
	iam:DeleteSigningCertificate

服务前缀	操作
	iam:DeleteUser
	iam:DeleteUserPermissionsBoundary
	iam:DeleteUserPolicy
	iam:DeleteVirtualMFADevice
	iam:DetachGroupPolicy
	iam:DetachRolePolicy
	iam:DetachUserPolicy
	iam:DisableOrganizationsRootCredentialsManagement
	iam:DisableOrganizationsRootSessions
	iam:DisableOutboundWebIdentityFederation
	iam:EnableMFADevice
	iam:EnableOrganizationsRootCredentialsManagement
	iam:EnableOrganizationsRootSessions
	iam:EnableOutboundWebIdentityFederation
	iam:GenerateCredentialReport
	iam:GenerateOrganizationsAccessReport
	iam:GenerateServiceLastAccessedDetails
	iam:GetAccessKeyLastUsed
	iam:GetAccountAuthorizationDetails
	iam:GetAccountEmailAddress
	iam:GetAccountName

服务前缀	操作
	iam:GetAccountPasswordPolicy
	iam:GetAccountSummary
	iam:GetCloudFrontPublicKey
	iam:GetContextKeysForCustomPolicy
	iam:GetContextKeysForPrincipalPolicy
	iam:GetCredentialReport
	iam:GetGroup
	iam:GetGroupPolicy
	iam:GetInstanceProfile
	iam:GetLoginProfile
	iam:GetMFADevice
	iam:GetOpenIDConnectProvider
	iam:GetOrganizationsAccessReport
	iam:GetOutboundWebIdentityFederationInfo
	iam:GetPolicy
	iam:GetPolicyVersion
	iam:GetRole
	iam:GetRolePolicy
	iam:GetSAMLProvider
	iam:GetSSHPublicKey
	iam:GetServerCertificate

服务前缀	操作
	iam:GetServiceLastAccessedDetails
	iam:GetServiceLastAccessedDetailsWithEntities
	iam:GetServiceLinkedRoleDeletionStatus
	iam:GetUser
	iam:GetUserPolicy
	iam:ListAccessKeys
	iam:ListAccountAliases
	iam:ListAttachedGroupPolicies
	iam:ListAttachedRolePolicies
	iam:ListAttachedUserPolicies
	iam:ListCloudFrontPublicKeys
	iam:ListDelegationRequests
	iam:ListEntitiesForPolicy
	iam:ListGroupPolicies
	iam:ListGroups
	iam:ListGroupsForUser
	iam:ListInstanceProfiles
	iam:ListInstanceProfilesForRole
	iam:ListMFADevices
	iam:ListOpenIDConnectProviders
	iam:ListOrganizationsFeatures

服务前缀	操作
	iam:ListPolicies
	iam:ListPoliciesGrantingServiceAccess
	iam:ListPolicyVersions
	iam:ListRolePolicies
	iam:ListRoles
	iam:ListSAMLProviders
	iam:ListSSHPublicKeys
	iam:ListSTSRegionalEndpointsStatus
	iam:ListServerCertificates
	iam:ListServiceSpecificCredentials
	iam:ListSigningCertificates
	iam:ListUserPolicies
	iam:ListUsers
	iam:ListVirtualMFADevices
	iam:PutGroupPolicy
	iam:PutRolePermissionsBoundary
	iam:PutRolePolicy
	iam:PutUserPermissionsBoundary
	iam:PutUserPolicy
	iam:RemoveClientIDFromOpenIDConnectProvider
	iam:RemoveRoleFromInstanceProfile

服务前缀	操作
	iam:RemoveUserFromGroup
	iam:ResetServiceSpecificCredential
	iam:ResyncMFADevice
	iam:SetDefaultPolicyVersion
	iam:SetSTSRegionalEndpointStatus
	iam:SetSecurityTokenServicePreferences
	iam:SimulateCustomPolicy
	iam:SimulatePrincipalPolicy
	iam:UpdateAccessKey
	iam:UpdateAccountEmailAddress
	iam:UpdateAccountName
	iam:UpdateAccountPasswordPolicy
	iam:UpdateAssumeRolePolicy
	iam:UpdateCloudFrontPublicKey
	iam:UpdateGroup
	iam:UpdateLoginProfile
	iam:UpdateOpenIDConnectProviderThumbprint
	iam:UpdateRole
	iam:UpdateRoleDescription
	iam:UpdateSAMLProvider
	iam:UpdateSSHPublicKey

服务前缀	操作
	iam:UpdateServerCertificate
	iam:UpdateServiceSpecificCredential
	iam:UpdateSigningCertificate
	iam:UpdateUser
	iam:UploadCloudFrontPublicKey
	iam:UploadSSHPublicKey
	iam:UploadServerCertificate
	iam:UploadSigningCertificate

服务前缀	操作
identitystore	identitystore:CreateGroup identitystore:CreateGroupMembership identitystore:CreateUser identitystore:DeleteGroup identitystore:DeleteGroupMembership identitystore:DeleteUser identitystore:DescribeGroup identitystore:DescribeGroupMembership identitystore:DescribeUser identitystore:GetGroupId identitystore:GetGroupMembershipId identitystore:GetUserId identitystore:IsMemberInGroups identitystore:ListGroupMemberships identitystore:ListGroupMembershipsForMember identitystore:ListGroups identitystore:ListUsers identitystore:UpdateGroup identitystore:UpdateUser

服务前缀	操作
imagebuilder	imagebuilder:CancelImageCreation imagebuilder:CancelLifecycleExecution imagebuilder:CreateComponent imagebuilder:CreateContainerRecipe imagebuilder:CreateDistributionConfiguration imagebuilder:CreateImage imagebuilder:CreateImagePipeline imagebuilder:CreateImageRecipe imagebuilder:CreateInfrastructureConfiguration imagebuilder:CreateLifecyclePolicy imagebuilder:CreateWorkflow imagebuilder:DeleteComponent imagebuilder:DeleteContainerRecipe imagebuilder:DeleteDistributionConfiguration imagebuilder:DeleteImage imagebuilder:DeleteImagePipeline imagebuilder:DeleteImageRecipe imagebuilder:DeleteInfrastructureConfiguration imagebuilder:DeleteLifecyclePolicy imagebuilder:DeleteWorkflow imagebuilder:DistributeImage

服务前缀	操作
	imagebuilder:GetComponentPolicy
	imagebuilder:GetContainerRecipePolicy
	imagebuilder:GetImagePolicy
	imagebuilder:GetImageRecipePolicy
	imagebuilder:GetLifecycleExecution
	imagebuilder:GetLifecyclePolicy
	imagebuilder:GetMarketplaceResource
	imagebuilder:GetWorkflowExecution
	imagebuilder:GetWorkflowStepExecution
	imagebuilder:ImportComponent
	imagebuilder:ImportDiskImage
	imagebuilder:ImportVmImage
	imagebuilder:ListComponentBuildVersions
	imagebuilder:ListComponents
	imagebuilder:ListContainerRecipes
	imagebuilder:ListDistributionConfigurations
	imagebuilder:ListImageBuildVersions
	imagebuilder:ListImagePackages
	imagebuilder:ListImagePipelineImages
	imagebuilder:ListImagePipelines
	imagebuilder:ListImageRecipes

服务前缀	操作
	imagebuilder:ListImageScanFindingAggregations
	imagebuilder:ListImageScanFindings
	imagebuilder:ListImages
	imagebuilder:ListInfrastructureConfigurations
	imagebuilder:ListLifecycleExecutionResources
	imagebuilder:ListLifecycleExecutions
	imagebuilder:ListLifecyclePolicies
	imagebuilder:ListWaitingWorkflowSteps
	imagebuilder:ListWorkflowExecutions
	imagebuilder:ListWorkflowStepExecutions
	imagebuilder:ListWorkflows
	imagebuilder:PutComponentPolicy
	imagebuilder:PutContainerRecipePolicy
	imagebuilder:PutImagePolicy
	imagebuilder:PutImageRecipePolicy
	imagebuilder:RetryImage
	imagebuilder:SendWorkflowStepAction
	imagebuilder:StartImagePipelineExecution
	imagebuilder:StartResourceStateUpdate
	imagebuilder:UpdateDistributionConfiguration
	imagebuilder:UpdateImagePipeline

服务前缀	操作
	imagebuilder:UpdateInfrastructureConfiguration

服务前缀	操作
inspector	inspector:AddAttributesToFindings inspector:CreateAssessmentTarget inspector:CreateAssessmentTemplate inspector:CreateExclusionsPreview inspector:CreateResourceGroup inspector:DeleteAssessmentRun inspector:DeleteAssessmentTarget inspector:DeleteAssessmentTemplate inspector:DescribeAssessmentRuns inspector:DescribeAssessmentTargets inspector:DescribeAssessmentTemplates inspector:DescribeCrossAccountAccessRole inspector:DescribeExclusions inspector:DescribeFindings inspector:DescribeResourceGroups inspector:DescribeRulesPackages inspector:GetAssessmentReport inspector:GetExclusionsPreview inspector:GetTelemetryMetadata inspector:ListAssessmentRunAgents inspector:ListAssessmentRuns

服务前缀	操作
	inspector:ListAssessmentTargets inspector:ListAssessmentTemplates inspector:ListEventSubscriptions inspector:ListExclusions inspector:ListFindings inspector:ListRulesPackages inspector:PreviewAgents inspector:RegisterCrossAccountAccessRole inspector:RemoveAttributesFromFindings inspector:StartAssessmentRun inspector:StopAssessmentRun inspector:SubscribeToEvent inspector:UnsubscribeFromEvent inspector:UpdateAssessmentTarget

服务前缀	操作
inspector2	inspector2:AssociateMember
	inspector2:BatchGetAccountStatus
	inspector2:BatchGetCodeSnippet
	inspector2:BatchGetFindingDetails
	inspector2:BatchGetFreeTrialInfo
	inspector2:BatchGetMemberEc2DeepInspectionStatus
	inspector2:BatchUpdateMemberEc2DeepInspectionStatus
	inspector2:CancelFindingsReport
	inspector2:CancelSbomExport
	inspector2:CreateCisScanConfiguration
	inspector2:CreateCodeSecurityIntegration
	inspector2:CreateFilter
	inspector2:CreateFindingsReport
	inspector2:CreateSbomExport
	inspector2:DeleteCisScanConfiguration
	inspector2:DeleteCodeSecurityIntegration
	inspector2:DeleteFilter
	inspector2:DescribeOrganizationConfiguration
	inspector2:Disable
	inspector2:DisableDelegatedAdminAccount
	inspector2:DisassociateMember

服务前缀	操作
	inspector2:Enable
	inspector2:EnableDelegatedAdminAccount
	inspector2:GetCisScanReport
	inspector2:GetCisScanResultDetails
	inspector2:GetClustersForImage
	inspector2:GetCodeSecurityIntegration
	inspector2:GetCodeSecurityScan
	inspector2:GetConfiguration
	inspector2:GetDelegatedAdminAccount
	inspector2:GetEc2DeepInspectionConfiguration
	inspector2:GetEncryptionKey
	inspector2:GetFindingsReportStatus
	inspector2:GetMember
	inspector2:GetSbomExport
	inspector2:ListAccountPermissions
	inspector2:ListCisScanConfigurations
	inspector2:ListCisScanResultsAggregatedByChecks
	inspector2:ListCisScanResultsAggregatedByTargetResource
	inspector2:ListCisScans
	inspector2:ListCodeSecurityIntegrations
	inspector2:ListCodeSecurityScanConfigurations

服务前缀	操作
	inspector2:ListCoverage
	inspector2:ListCoverageStatistics
	inspector2:ListDelegatedAdminAccounts
	inspector2:ListFilters
	inspector2:ListFindingAggregations
	inspector2:ListFindings
	inspector2:ListMembers
	inspector2:ListUsageTotals
	inspector2:ResetEncryptionKey
	inspector2:SearchVulnerabilities
	inspector2:SendCisSessionHealth
	inspector2:SendCisSessionTelemetry
	inspector2:StartCisSession
	inspector2:StartCodeSecurityScan
	inspector2:StopCisSession
	inspector2:UpdateCisScanConfiguration
	inspector2:UpdateCodeSecurityIntegration
	inspector2:UpdateConfiguration
	inspector2:UpdateEc2DeepInspectionConfiguration
	inspector2:UpdateEncryptionKey
	inspector2:UpdateFilter

服务前缀	操作
	inspector2:UpdateOrgEc2DeepInspectionConfiguration inspector2:UpdateOrganizationConfiguration

服务前缀	操作
iot	iot:AcceptCertificateTransfer iot:AddThingToBillingGroup iot:AddThingToThingGroup iot:AssociateSbomWithPackageVersion iot:AssociateTargetsWithJob iot:AttachPolicy iot:AttachPrincipalPolicy iot:AttachSecurityProfile iot:AttachThingPrincipal iot:CancelAuditMitigationActionsTask iot:CancelAuditTask iot:CancelCertificateTransfer iot:CancelDetectMitigationActionsTask iot:CancelJob iot:CancelJobExecution iot:ClearDefaultAuthorizer iot:ConfirmTopicRuleDestination iot:CreateAuditSuppression iot:CreateAuthorizer iot:CreateBillingGroup iot:CreateCertificateFromCsr

服务前缀	操作
	iot:CreateCertificateProvider iot:CreateCommand iot:CreateCustomMetric iot:CreateDimension iot:CreateDomainConfiguration iot:CreateDynamicThingGroup iot:CreateFleetMetric iot:CreateJob iot:CreateJobTemplate iot:CreateKeysAndCertificate iot:CreateMitigationAction iot:CreateOTAUpdate iot:CreatePackage iot:CreatePackageVersion iot:CreatePolicy iot:CreatePolicyVersion iot:CreateProvisioningClaim iot:CreateProvisioningTemplate iot:CreateProvisioningTemplateVersion iot:CreateRoleAlias iot:CreateScheduledAudit

服务前缀	操作
	iot:CreateSecurityProfile iot:CreateStream iot:CreateThing iot:CreateThingGroup iot:CreateThingType iot:CreateTopicRule iot:CreateTopicRuleDestination iot:DeleteAccountAuditConfiguration iot:DeleteAuditSuppression iot:DeleteAuthorizer iot:DeleteBillingGroup iot:DeleteCACertificate iot:DeleteCertificate iot:DeleteCertificateProvider iot:DeleteCommand iot:DeleteCustomMetric iot:DeleteDimension iot:DeleteDomainConfiguration iot:DeleteDynamicThingGroup iot:DeleteFleetMetric iot:DeleteJob

服务前缀	操作
	iot:DeleteJobExecution iot:DeleteJobTemplate iot:DeleteMitigationAction iot:DeleteOTAUpdate iot:DeletePackage iot:DeletePackageVersion iot:DeletePolicy iot:DeletePolicyVersion iot:DeleteProvisioningTemplate iot:DeleteProvisioningTemplateVersion iot:DeleteRegistrationCode iot:DeleteRoleAlias iot:DeleteScheduledAudit iot:DeleteSecurityProfile iot:DeleteStream iot:DeleteThing iot:DeleteThingGroup iot:DeleteThingType iot:DeleteTopicRule iot:DeleteTopicRuleDestination iot:DeleteV2LoggingLevel

服务前缀	操作
	iot:DeprecateThingType iot:DescribeAccountAuditConfiguration iot:DescribeAuditFinding iot:DescribeAuditMitigationActionsTask iot:DescribeAuditSuppression iot:DescribeAuditTask iot:DescribeAuthorizer iot:DescribeBillingGroup iot:DescribeCACertificate iot:DescribeCertificate iot:DescribeCertificateProvider iot:DescribeCustomMetric iot:DescribeDefaultAuthorizer iot:DescribeDetectMitigationActionsTask iot:DescribeDimension iot:DescribeDomainConfiguration iot:DescribeEncryptionConfiguration iot:DescribeEndpoint iot:DescribeEventConfigurations iot:DescribeFleetMetric iot:DescribeIndex

服务前缀	操作
	iot:DescribeJob iot:DescribeJobExecution iot:DescribeJobTemplate iot:DescribeManagedJobTemplate iot:DescribeMitigationAction iot:DescribeProvisioningTemplate iot:DescribeProvisioningTemplateVersion iot:DescribeRoleAlias iot:DescribeScheduledAudit iot:DescribeSecurityProfile iot:DescribeStream iot:DescribeThing iot:DescribeThingGroup iot:DescribeThingRegistrationTask iot:DescribeThingType iot:DetachPolicy iot:DetachPrincipalPolicy iot:DetachSecurityProfile iot:DetachThingPrincipal iot:DisableTopicRule iot:DisassociateSbomFromPackageVersion

服务前缀	操作
	iot:EnableTopicRule iot:GetBehaviorModelTrainingSummaries iot:GetBucketsAggregation iot:GetCardinality iot:GetCommand iot:GetEffectivePolicies iot:GetJobDocument iot:GetLoggingOptions iot:GetOTAUpdate iot:GetPackage iot:GetPackageConfiguration iot:GetPackageVersion iot:GetPercentiles iot:GetPolicy iot:GetPolicyVersion iot:GetRegistrationCode iot:GetStatistics iot:GetThingConnectivityData iot:GetTopicRule iot:GetTopicRuleDestination iot:GetV2LoggingOptions

服务前缀	操作
	iot:ListActiveViolations iot:ListAttachedPolicies iot:ListAuditFindings iot:ListAuditMitigationActionsExecutions iot:ListAuditMitigationActionsTasks iot:ListAuditSuppressions iot:ListAuditTasks iot:ListAuthorizers iot:ListBillingGroups iot:ListCACertificates iot:ListCertificateProviders iot:ListCertificates iot:ListCertificatesByCA iot:ListCommands iot:ListCustomMetrics iot:ListDetectMitigationActionsExecutions iot:ListDetectMitigationActionsTasks iot:ListDimensions iot:ListDomainConfigurations iot:ListFleetMetrics iot:ListIndices

服务前缀	操作
	iot:ListJobExecutionsForJob iot:ListJobExecutionsForThing iot:ListJobTemplates iot:ListJobs iot:ListManagedJobTemplates iot:ListMetricValues iot:ListMitigationActions iot:ListOTAUpdates iot:ListOutgoingCertificates iot:ListPackageVersions iot:ListPackages iot:ListPolicies iot:ListPolicyPrincipals iot:ListPolicyVersions iot:ListPrincipalPolicies iot:ListPrincipalThings iot:ListPrincipalThingsV2 iot:ListProvisioningTemplateVersions iot:ListProvisioningTemplates iot:ListRelatedResourcesForAuditFinding iot:ListRoleAliases

服务前缀	操作
	iot:ListSbomValidationResults iot:ListScheduledAudits iot:ListSecurityProfiles iot:ListSecurityProfilesForTarget iot:ListStreams iot:ListTargetsForPolicy iot:ListTargetsForSecurityProfile iot:ListThingGroups iot:ListThingGroupsForThing iot:ListThingPrincipals iot:ListThingPrincipalsV2 iot:ListThingRegistrationTaskReports iot:ListThingRegistrationTasks iot:ListThingTypes iot:ListThings iot:ListThingsInBillingGroup iot:ListThingsInThingGroup iot:ListTopicRuleDestinations iot:ListTopicRules iot:ListV2LoggingLevels iot:ListViolationEvents

服务前缀	操作
	iot:PutVerificationStateOnViolation iot:RegisterCACertificate iot:RegisterCertificate iot:RegisterCertificateWithoutCA iot:RegisterThing iot:RejectCertificateTransfer iot:RemoveThingFromBillingGroup iot:RemoveThingFromThingGroup iot:ReplaceTopicRule iot:SearchIndex iot:SetDefaultAuthorizer iot:SetDefaultPolicyVersion iot:SetLoggingOptions iot:SetV2LoggingLevel iot:SetV2LoggingOptions iot:StartAuditMitigationActionsTask iot:StartDetectMitigationActionsTask iot:StartOnDemandAuditTask iot:StartThingRegistrationTask iot:StopThingRegistrationTask iot:TestAuthorization

服务前缀	操作
	iot:TestInvokeAuthorizer iot:TransferCertificate iot:UpdateAccountAuditConfiguration iot:UpdateAuditSuppression iot:UpdateAuthorizer iot:UpdateBillingGroup iot:UpdateCACertificate iot:UpdateCertificate iot:UpdateCertificateProvider iot:UpdateCommand iot:UpdateCustomMetric iot:UpdateDimension iot:UpdateDomainConfiguration iot:UpdateDynamicThingGroup iot:UpdateEncryptionConfiguration iot:UpdateEventConfigurations iot:UpdateFleetMetric iot:UpdateIndexingConfiguration iot:UpdateJob iot:UpdateMitigationAction iot:UpdatePackage

服务前缀	操作
	iot:UpdatePackageConfiguration iot:UpdatePackageVersion iot:UpdateProvisioningTemplate iot:UpdateRoleAlias iot:UpdateScheduledAudit iot:UpdateSecurityProfile iot:UpdateStream iot:UpdateThing iot:UpdateThingGroup iot:UpdateThingGroupsForThing iot:UpdateThingType iot:UpdateTopicRuleDestination iot:ValidateSecurityProfileBehaviors

服务前缀	操作
iotanalytics	iotanalytics:CancelPipelineReprocessing
	iotanalytics:CreateChannel
	iotanalytics:CreateDataset
	iotanalytics:CreateDatasetContent
	iotanalytics:CreateDatastore
	iotanalytics:CreatePipeline
	iotanalytics:DeleteChannel
	iotanalytics:DeleteDataset
	iotanalytics:DeleteDatasetContent
	iotanalytics:DeleteDatastore
	iotanalytics:DeletePipeline
	iotanalytics:DescribeChannel
	iotanalytics:DescribeDataset
	iotanalytics:DescribeDatastore
	iotanalytics:DescribeLoggingOptions
	iotanalytics:DescribePipeline
	iotanalytics:GetDatasetContent
	iotanalytics:ListChannels
	iotanalytics:ListDatasetContents
	iotanalytics:ListDatasets
	iotanalytics:ListDatastores

服务前缀	操作
	iotanalytics:ListPipelines iotanalytics:PutLoggingOptions iotanalytics:RunPipelineActivity iotanalytics:SampleChannelData iotanalytics:StartPipelineReprocessing iotanalytics:UpdateChannel iotanalytics:UpdateDataset iotanalytics:UpdateDatastore iotanalytics:UpdatePipeline
iotdeviceadvisor	iotdeviceadvisor:CreateSuiteDefinition iotdeviceadvisor:DeleteSuiteDefinition iotdeviceadvisor:GetEndpoint iotdeviceadvisor:GetSuiteDefinition iotdeviceadvisor:GetSuiteRun iotdeviceadvisor:GetSuiteRunReport iotdeviceadvisor:ListSuiteDefinitions iotdeviceadvisor:ListSuiteRuns iotdeviceadvisor:StartSuiteRun iotdeviceadvisor:StopSuiteRun iotdeviceadvisor:UpdateSuiteDefinition

服务前缀	操作
iotevents	iotevents:BatchAcknowledgeAlarm
	iotevents:BatchDeleteDetector
	iotevents:BatchDisableAlarm
	iotevents:BatchEnableAlarm
	iotevents:BatchResetAlarm
	iotevents:BatchSnoozeAlarm
	iotevents:BatchUpdateDetector
	iotevents:CreateAlarmModel
	iotevents:CreateDetectorModel
	iotevents:CreateInput
	iotevents:DeleteAlarmModel
	iotevents:DeleteDetectorModel
	iotevents:DeleteInput
	iotevents:DescribeAlarm
	iotevents:DescribeAlarmModel
	iotevents:DescribeDetector
	iotevents:DescribeDetectorModel
	iotevents:DescribeDetectorModelAnalysis
	iotevents:DescribeInput
	iotevents:DescribeLoggingOptions
	iotevents:GetDetectorModelAnalysisResults

服务前缀	操作
	iotevents:ListAlarmModelVersions iotevents:ListAlarmModels iotevents:ListAlarms iotevents:ListDetectorModelVersions iotevents:ListDetectorModels iotevents:ListDetectors iotevents:ListInputRoutings iotevents:ListInputs iotevents:PutLoggingOptions iotevents:StartDetectorModelAnalysis iotevents:UpdateAlarmModel iotevents:UpdateDetectorModel iotevents:UpdateInput
iotfleethub	iotfleethub:CreateApplication iotfleethub:DeleteApplication iotfleethub:DescribeApplication iotfleethub:ListApplications iotfleethub:UpdateApplication

服务前缀	操作
iotsitewise	iotsitewise:AssociateAssets
	iotsitewise:AssociateTimeSeriesToAssetProperty
	iotsitewise:BatchAssociateProjectAssets
	iotsitewise:BatchDisassociateProjectAssets
	iotsitewise:CreateAccessPolicy
	iotsitewise:CreateAsset
	iotsitewise:CreateAssetModel
	iotsitewise:CreateAssetModelCompositeModel
	iotsitewise:CreateBulkImportJob
	iotsitewise:CreateComputationModel
	iotsitewise:CreateDashboard
	iotsitewise:CreateDataset
	iotsitewise:CreateGateway
	iotsitewise:CreatePortal
	iotsitewise:CreateProject
	iotsitewise:DeleteAccessPolicy
	iotsitewise:DeleteAsset
	iotsitewise:DeleteAssetModel
	iotsitewise:DeleteAssetModelCompositeModel
	iotsitewise:DeleteComputationModel
	iotsitewise:DeleteDashboard

服务前缀	操作
	iotsitewise:DeleteDataset
	iotsitewise:DeleteGateway
	iotsitewise:DeletePortal
	iotsitewise:DeleteProject
	iotsitewise:DeleteTimeSeries
	iotsitewise:DescribeAccessPolicy
	iotsitewise:DescribeAsset
	iotsitewise:DescribeAssetCompositeModel
	iotsitewise:DescribeAssetModel
	iotsitewise:DescribeAssetModelCompositeModel
	iotsitewise:DescribeAssetModelInterfaceRelationship
	iotsitewise:DescribeAssetProperty
	iotsitewise:DescribeBulkImportJob
	iotsitewise:DescribeComputationModel
	iotsitewise:DescribeComputationModelExecutionSummary
	iotsitewise:DescribeDashboard
	iotsitewise:DescribeDataset
	iotsitewise:DescribeDefaultEncryptionConfiguration
	iotsitewise:DescribeExecution
	iotsitewise:DescribeGateway
	iotsitewise:DescribeGatewayCapabilityConfiguration

服务前缀	操作
	iotsitewise:DescribeLoggingOptions
	iotsitewise:DescribePortal
	iotsitewise:DescribeProject
	iotsitewise:DescribeStorageConfiguration
	iotsitewise:DescribeTimeSeries
	iotsitewise:DisassociateAssets
	iotsitewise:DisassociateTimeSeriesFromAssetProperty
	iotsitewise:ExecuteAction
	iotsitewise:ExecuteQuery
	iotsitewise:ListAccessPolicies
	iotsitewise:ListActions
	iotsitewise:ListAssetModelCompositeModels
	iotsitewise:ListAssetModelProperties
	iotsitewise:ListAssetModels
	iotsitewise:ListAssetProperties
	iotsitewise:ListAssetRelationships
	iotsitewise:ListAssets
	iotsitewise:ListAssociatedAssets
	iotsitewise:ListBulkImportJobs
	iotsitewise:ListCompositionRelationships
	iotsitewise:ListComputationModelDataBindingUsages

服务前缀	操作
	iotsitewise:ListComputationModelResolveToResources
	iotsitewise:ListComputationModels
	iotsitewise:ListDashboards
	iotsitewise:ListDatasets
	iotsitewise:ListExecutions
	iotsitewise:ListGateways
	iotsitewise:ListInterfaceRelationships
	iotsitewise:ListPortals
	iotsitewise:ListProjectAssets
	iotsitewise:ListProjects
	iotsitewise:ListTimeSeries
	iotsitewise:PutDefaultEncryptionConfiguration
	iotsitewise:PutLoggingOptions
	iotsitewise:PutStorageConfiguration
	iotsitewise:UpdateAccessPolicy
	iotsitewise:UpdateAsset
	iotsitewise:UpdateAssetModel
	iotsitewise:UpdateAssetModelCompositeModel
	iotsitewise:UpdateAssetProperty
	iotsitewise:UpdateComputationModel
	iotsitewise:UpdateDashboard

服务前缀	操作
	iotsitewise:UpdateDataset iotsitewise:UpdateGateway iotsitewise:UpdateGatewayCapabilityConfiguration iotsitewise:UpdatePortal iotsitewise:UpdateProject

服务前缀	操作
iottwinmaker	iottwinmaker:CancelMetadataTransferJob iottwinmaker:CreateComponentType iottwinmaker:CreateEntity iottwinmaker:CreateMetadataTransferJob iottwinmaker:CreateScene iottwinmaker:CreateSyncJob iottwinmaker:CreateWorkspace iottwinmaker:DeleteComponentType iottwinmaker:DeleteEntity iottwinmaker:DeleteScene iottwinmaker:DeleteSyncJob iottwinmaker:DeleteWorkspace iottwinmaker:ExecuteQuery iottwinmaker:GetMetadataTransferJob iottwinmaker:GetPricingPlan iottwinmaker:GetScene iottwinmaker:GetSyncJob iottwinmaker:ListComponentTypes iottwinmaker:ListComponents iottwinmaker:ListEntities iottwinmaker:ListMetadataTransferJobs

服务前缀	操作
	iottwinmaker:ListProperties iottwinmaker:ListScenes iottwinmaker:ListSyncJobs iottwinmaker:ListSyncResources iottwinmaker:ListWorkspaces iottwinmaker:UpdateComponentType iottwinmaker:UpdateEntity iottwinmaker:UpdatePricingPlan iottwinmaker:UpdateScene iottwinmaker:UpdateWorkspace

服务前缀	操作
iotwireless	iotwireless:AssociateAwsAccountWithPartnerAccount
	iotwireless:AssociateMulticastGroupWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithMulticastGroup
	iotwireless:AssociateWirelessDeviceWithThing
	iotwireless:AssociateWirelessGatewayWithCertificate
	iotwireless:AssociateWirelessGatewayWithThing
	iotwireless:CancelMulticastGroupSession
	iotwireless:CreateDestination
	iotwireless:CreateDeviceProfile
	iotwireless:CreateFuotaTask
	iotwireless:CreateMulticastGroup
	iotwireless:CreateNetworkAnalyzerConfiguration
	iotwireless:CreateServiceProfile
	iotwireless:CreateWirelessDevice
	iotwireless:CreateWirelessGateway
	iotwireless:CreateWirelessGatewayTask
	iotwireless:CreateWirelessGatewayTaskDefinition
	iotwireless:DeleteDestination
	iotwireless:DeleteDeviceProfile
	iotwireless:DeleteFuotaTask

服务前缀	操作
	iotwireless:DeleteMulticastGroup
	iotwireless:DeleteNetworkAnalyzerConfiguration
	iotwireless:DeleteQueuedMessages
	iotwireless:DeleteServiceProfile
	iotwireless:DeleteWirelessDevice
	iotwireless:DeleteWirelessDeviceImportTask
	iotwireless:DeleteWirelessGateway
	iotwireless:DeleteWirelessGatewayTask
	iotwireless:DeleteWirelessGatewayTaskDefinition
	iotwireless:DeregisterWirelessDevice
	iotwireless:DisassociateAwsAccountFromPartnerAccount
	iotwireless:DisassociateMulticastGroupFromFuotaTask
	iotwireless:DisassociateWirelessDeviceFromFuotaTask
	iotwireless:DisassociateWirelessDeviceFromMulticastGroup
	iotwireless:DisassociateWirelessDeviceFromThing
	iotwireless:DisassociateWirelessGatewayFromCertificate
	iotwireless:DisassociateWirelessGatewayFromThing
	iotwireless:GetDestination
	iotwireless:GetDeviceProfile
	iotwireless:GetEventConfigurationByResourceTypes
	iotwireless:GetFuotaTask

服务前缀	操作
	iotwireless:GetLogLevelsByResourceTypes
	iotwireless:GetMetricConfiguration
	iotwireless:GetMetrics
	iotwireless:GetMulticastGroup
	iotwireless:GetMulticastGroupSession
	iotwireless:GetNetworkAnalyzerConfiguration
	iotwireless:GetPartnerAccount
	iotwireless:GetPosition
	iotwireless:GetPositionConfiguration
	iotwireless:GetPositionEstimate
	iotwireless:GetResourceEventConfiguration
	iotwireless:GetResourceLogLevel
	iotwireless:GetResourcePosition
	iotwireless:GetServiceEndpoint
	iotwireless:GetServiceProfile
	iotwireless:GetWirelessDevice
	iotwireless:GetWirelessDeviceImportTask
	iotwireless:GetWirelessDeviceStatistics
	iotwireless:GetWirelessGateway
	iotwireless:GetWirelessGatewayCertificate
	iotwireless:GetWirelessGatewayFirmwareInformation

服务前缀	操作
	iotwireless:GetWirelessGatewayStatistics
	iotwireless:GetWirelessGatewayTask
	iotwireless:GetWirelessGatewayTaskDefinition
	iotwireless:ListDestinations
	iotwireless:ListDeviceProfiles
	iotwireless:ListDevicesForWirelessDeviceImportTask
	iotwireless:ListEventConfigurations
	iotwireless:ListFuotaTasks
	iotwireless:ListMulticastGroups
	iotwireless:ListMulticastGroupsByFuotaTask
	iotwireless:ListNetworkAnalyzerConfigurations
	iotwireless:ListPartnerAccounts
	iotwireless:ListPositionConfigurations
	iotwireless:ListQueuedMessages
	iotwireless:ListServiceProfiles
	iotwireless:ListWirelessDeviceImportTasks
	iotwireless:ListWirelessDevices
	iotwireless:ListWirelessGatewayTaskDefinitions
	iotwireless:ListWirelessGateways
	iotwireless:PutPositionConfiguration
	iotwireless:PutResourceLogLevel

服务前缀	操作
	iotwireless:ResetAllResourceLogLevels
	iotwireless:ResetResourceLogLevel
	iotwireless:SendDataToMulticastGroup
	iotwireless:SendDataToWirelessDevice
	iotwireless:StartBulkAssociateWirelessDeviceWithMulticastGroup
	iotwireless:StartBulkDisassociateWirelessDeviceFromMulticastGroup
	iotwireless:StartFuotaTask
	iotwireless:StartMulticastGroupSession
	iotwireless:StartNetworkAnalyzerStream
	iotwireless:StartSingleWirelessDeviceImportTask
	iotwireless:StartWirelessDeviceImportTask
	iotwireless:TestWirelessDevice
	iotwireless:UpdateDestination
	iotwireless:UpdateEventConfigurationByResourceTypes
	iotwireless:UpdateFuotaTask
	iotwireless:UpdateLogLevelsByResourceTypes
	iotwireless:UpdateMetricConfiguration
	iotwireless:UpdateMulticastGroup
	iotwireless:UpdateNetworkAnalyzerConfiguration
	iotwireless:UpdatePartnerAccount

服务前缀	操作
	iotwireless:UpdatePosition
	iotwireless:UpdateResourceEventConfiguration
	iotwireless:UpdateResourcePosition
	iotwireless:UpdateWirelessDevice
	iotwireless:UpdateWirelessDeviceImportTask
	iotwireless:UpdateWirelessGateway

服务前缀	操作
ivs	ivs:BatchGetChannel ivs:BatchGetStreamKey ivs:BatchStartViewerSessionRevocation ivs:CreateChannel ivs:CreateEncoderConfiguration ivs:CreateIngestConfiguration ivs:CreateParticipantToken ivs:CreatePlaybackRestrictionPolicy ivs:CreateRecordingConfiguration ivs:CreateStorageConfiguration ivs:CreateStreamKey ivs:DeleteChannel ivs:DeleteEncoderConfiguration ivs:DeleteIngestConfiguration ivs:DeletePlaybackKeyPair ivs:DeletePlaybackRestrictionPolicy ivs:DeletePublicKey ivs:DeleteRecordingConfiguration ivs:DeleteStorageConfiguration ivs:DeleteStreamKey ivs:DisconnectParticipant

服务前缀	操作
	ivs:GetChannel ivs:GetComposition ivs:GetEncoderConfiguration ivs:GetIngestConfiguration ivs:GetParticipant ivs:GetPlaybackKeyPair ivs:GetPlaybackRestrictionPolicy ivs:GetPublicKey ivs:GetRecordingConfiguration ivs:GetStorageConfiguration ivs:GetStream ivs:GetStreamKey ivs:GetStreamSession ivs:ImportPlaybackKeyPair ivs:ImportPublicKey ivs:ListChannels ivs:ListCompositions ivs:ListEncoderConfigurations ivs:ListIngestConfigurations ivs:ListParticipantEvents ivs:ListParticipantReplicas

服务前缀	操作
	ivs:ListParticipants ivs:ListPlaybackKeyPairs ivs:ListPlaybackRestrictionPolicies ivs:ListPublicKeys ivs:ListRecordingConfigurations ivs:ListStorageConfigurations ivs:ListStreamKeys ivs:ListStreamSessions ivs:ListStreams ivs:PutMetadata ivs:StartComposition ivs:StartViewerSessionRevocation ivs:StopComposition ivs:StopStream ivs:UpdateChannel ivs:UpdateIngestConfiguration ivs:UpdatePlaybackRestrictionPolicy

服务前缀	操作
ivschat	ivschat:CreateChatToken ivschat:CreateLoggingConfiguration ivschat:CreateRoom ivschat:DeleteLoggingConfiguration ivschat:DeleteMessage ivschat:DeleteRoom ivschat:DisconnectUser ivschat:GetLoggingConfiguration ivschat:GetRoom ivschat:ListLoggingConfigurations ivschat:ListRooms ivschat:SendEvent ivschat:UpdateLoggingConfiguration ivschat:UpdateRoom

服务前缀	操作
kafka	kafka:BatchAssociateScramSecret kafka:BatchDisassociateScramSecret kafka:CreateCluster kafka:CreateClusterV2 kafka:CreateConfiguration kafka:CreateReplicator kafka:CreateVpcConnection kafka:DeleteCluster kafka:DeleteClusterPolicy kafka:DeleteConfiguration kafka:DeleteReplicator kafka:DeleteVpcConnection kafka:DescribeCluster kafka:DescribeClusterOperation kafka:DescribeClusterOperationV2 kafka:DescribeClusterV2 kafka:DescribeConfiguration kafka:DescribeConfigurationRevision kafka:DescribeVpcConnection kafka:GetBootstrapBrokers kafka:GetClusterPolicy

服务前缀	操作
	kafka:GetCompatibleKafkaVersions kafka:ListClientVpcConnections kafka:ListClusterOperations kafka:ListClusterOperationsV2 kafka:ListClusters kafka:ListClustersV2 kafka:ListConfigurationRevisions kafka:ListConfigurations kafka:ListKafkaVersions kafka:ListNodes kafka:ListReplicators kafka:ListScramSecrets kafka:ListVpcConnections kafka:PutClusterPolicy kafka:RebootBroker kafka:RejectClientVpcConnection kafka:UpdateBrokerCount kafka:UpdateBrokerStorage kafka:UpdateBrokerType kafka:UpdateClusterConfiguration kafka:UpdateClusterKafkaVersion

服务前缀	操作
	kafka:UpdateConfiguration kafka:UpdateConnectivity kafka:UpdateMonitoring kafka:UpdateRebalancing kafka:UpdateReplicationInfo kafka:UpdateSecurity kafka:UpdateStorage
kafkaconnect	kafkaconnect:CreateConnector kafkaconnect:CreateCustomPlugin kafkaconnect:CreateWorkerConfiguration kafkaconnect:DeleteConnector kafkaconnect:DeleteCustomPlugin kafkaconnect:DeleteWorkerConfiguration kafkaconnect:DescribeConnector kafkaconnect:DescribeCustomPlugin kafkaconnect:DescribeWorkerConfiguration kafkaconnect:ListConnectorOperations kafkaconnect:ListConnectors kafkaconnect:ListCustomPlugins kafkaconnect:ListWorkerConfigurations kafkaconnect:UpdateConnector

服务前缀	操作
kendra	kendra:AssociateEntitiesToExperience kendra:AssociatePersonasToEntities kendra:BatchDeleteDocument kendra:BatchDeleteFeaturedResultsSet kendra:BatchGetDocumentStatus kendra:BatchPutDocument kendra:ClearQuerySuggestions kendra:CreateAccessControlConfiguration kendra:CreateDataSource kendra:CreateExperience kendra:CreateFaq kendra:CreateFeaturedResultsSet kendra:CreateIndex kendra:CreateQuerySuggestionsBlockList kendra:CreateThesaurus kendra:DeleteDataSource kendra:DeleteExperience kendra:DeleteFaq kendra:DeleteIndex kendra:DeletePrincipalMapping kendra:DeleteQuerySuggestionsBlockList

服务前缀	操作
	kendra:DeleteThesaurus kendra:DescribeAccessControlConfiguration kendra:DescribeDataSource kendra:DescribeExperience kendra:DescribeFaq kendra:DescribeFeaturedResultsSet kendra:DescribeIndex kendra:DescribePrincipalMapping kendra:DescribeQuerySuggestionsBlockList kendra:DescribeQuerySuggestionsConfig kendra:DescribeThesaurus kendra:DisassociateEntitiesFromExperience kendra:DisassociatePersonasFromEntities kendra:GetQuerySuggestions kendra:GetSnapshots kendra:ListAccessControlConfigurations kendra:ListDataSourceSyncJobs kendra:ListDataSources kendra:ListEntityPersonas kendra:ListExperienceEntities kendra:ListExperiences

服务前缀	操作
	kendra:ListFaqs kendra:ListFeaturedResultsSets kendra:ListGroupsOlderThanOrderingId kendra:ListIndices kendra:ListQuerySuggestionsBlockLists kendra:ListThesauri kendra:PutPrincipalMapping kendra:Query kendra:Retrieve kendra:StartDataSourceSyncJob kendra:StopDataSourceSyncJob kendra:SubmitFeedback kendra:UpdateDataSource kendra:UpdateExperience kendra:UpdateFeaturedResultsSet kendra:UpdateIndex kendra:UpdateQuerySuggestionsBlockList kendra:UpdateQuerySuggestionsConfig kendra:UpdateThesaurus

服务前缀	操作
kinesis	kinesis:CreateStream
	kinesis:DecreaseStreamRetentionPeriod
	kinesis:DeleteStream
	kinesis:DeregisterStreamConsumer
	kinesis:DescribeAccountSettings
	kinesis:DescribeLimits
	kinesis:DescribeStream
	kinesis:DescribeStreamConsumer
	kinesis:DescribeStreamSummary
	kinesis:DisableEnhancedMonitoring
	kinesis:EnableEnhancedMonitoring
	kinesis:IncreaseStreamRetentionPeriod
	kinesis:ListShards
	kinesis:ListStreamConsumers
	kinesis:ListStreams
	kinesis:MergeShards
	kinesis:RegisterStreamConsumer
	kinesis:SplitShard
	kinesis:StartStreamEncryption
	kinesis:StopStreamEncryption
	kinesis:UpdateAccountSettings

服务前缀	操作
	kinesis:UpdateShardCount kinesis:UpdateStreamMode

服务前缀	操作
kinesisanalytics	kinesisanalytics:AddApplicationCloudWatchLoggingOption kinesisanalytics:AddApplicationInput kinesisanalytics:AddApplicationInputProcessingConfiguration kinesisanalytics:AddApplicationOutput kinesisanalytics:AddApplicationReferenceDataSource kinesisanalytics:AddApplicationVpcConfiguration kinesisanalytics:CreateApplication kinesisanalytics:CreateApplicationPresignedUrl kinesisanalytics:CreateApplicationSnapshot kinesisanalytics:DeleteApplication kinesisanalytics:DeleteApplicationCloudWatchLoggingOption kinesisanalytics:DeleteApplicationInputProcessingConfiguration kinesisanalytics:DeleteApplicationOutput kinesisanalytics:DeleteApplicationReferenceDataSource kinesisanalytics:DeleteApplicationSnapshot kinesisanalytics:DeleteApplicationVpcConfiguration kinesisanalytics:DescribeApplication kinesisanalytics:DescribeApplicationOperation kinesisanalytics:DescribeApplicationSnapshot kinesisanalytics:DescribeApplicationVersion kinesisanalytics:DiscoverInputSchema

服务前缀	操作
	kinesisanalytics:ListApplicationOperations kinesisanalytics:ListApplicationSnapshots kinesisanalytics:ListApplicationVersions kinesisanalytics:ListApplications kinesisanalytics:RollbackApplication kinesisanalytics:StartApplication kinesisanalytics:StopApplication kinesisanalytics:UpdateApplication kinesisanalytics:UpdateApplicationMaintenanceConfiguration

服务前缀	操作
kms	kms:CancelKeyDeletion
	kms:ConnectCustomKeyStore
	kms:CreateAlias
	kms:CreateCustomKeyStore
	kms:CreateGrant
	kms:CreateKey
	kms:Decrypt
	kms:DeleteAlias
	kms:DeleteCustomKeyStore
	kms:DeleteImportedKeyMaterial
	kms:DeriveSharedSecret
	kms:DescribeCustomKeyStores
	kms:DescribeKey
	kms:DisableKey
	kms:DisableKeyRotation
	kms:DisconnectCustomKeyStore
	kms:EnableKey
	kms:EnableKeyRotation
	kms:Encrypt
	kms:GenerateDataKey
	kms:GenerateDataKeyPair

服务前缀	操作
	kms:GenerateDataKeyPairWithoutPlaintext
	kms:GenerateDataKeyWithoutPlaintext
	kms:GenerateMac
	kms:GenerateRandom
	kms:GetKeyPolicy
	kms:GetKeyRotationStatus
	kms:GetParametersForImport
	kms:GetPublicKey
	kms:ImportKeyMaterial
	kms:ListAliases
	kms:ListGrants
	kms:ListKeyPolicies
	kms:ListKeyRotations
	kms:ListKeys
	kms:ListRetirableGrants
	kms:ReplicateKey
	kms:RetireGrant
	kms:RevokeGrant
	kms:RotateKeyOnDemand
	kms:ScheduleKeyDeletion
	kms:Sign

服务前缀	操作
	kms:UpdateAlias kms:UpdateCustomKeyStore kms:UpdateKeyDescription kms:UpdatePrimaryRegion kms:Verify kms:VerifyMac

服务前缀	操作
lambda	lambda:AddLayerVersionPermission lambda:AddPermission lambda:CreateAlias lambda:CreateCodeSigningConfig lambda:CreateEventSourceMapping lambda:CreateFunction lambda:CreateFunctionUrlConfig lambda:DeleteAlias lambda:DeleteCapacityProvider lambda:DeleteCodeSigningConfig lambda:DeleteEventSourceMapping lambda:DeleteFunction lambda:DeleteFunctionCodeSigningConfig lambda:DeleteFunctionConcurrency lambda:DeleteFunctionEventInvokeConfig lambda:DeleteFunctionUrlConfig lambda:DeleteLayerVersion lambda:DeleteProvisionedConcurrencyConfig lambda:GetAccountSettings lambda:GetAlias lambda:GetCapacityProvider

服务前缀	操作
	lambda:GetCodeSigningConfig lambda:GetEventSourceMapping lambda:GetFunction lambda:GetFunctionCodeSigningConfig lambda:GetFunctionConcurrency lambda:GetFunctionConfiguration lambda:GetFunctionEventInvokeConfig lambda:GetFunctionRecursionConfig lambda:GetFunctionScalingConfig lambda:GetFunctionUrlConfig lambda:GetLayerVersion lambda:GetLayerVersionPolicy lambda:GetPolicy lambda:GetProvisionedConcurrencyConfig lambda:GetRuntimeManagementConfig lambda:ListAliases lambda:ListCapacityProviders lambda:ListCodeSigningConfigs lambda:ListDurableExecutionsByFunction lambda:ListEventSourceMappings lambda:ListFunctionEventInvokeConfigs

服务前缀	操作
	lambda:ListFunctionUrlConfigs lambda:ListFunctions lambda:ListFunctionsByCodeSigningConfig lambda:ListLayerVersions lambda:ListLayers lambda:ListProvisionedConcurrencyConfigs lambda:ListVersionsByFunction lambda:PublishLayerVersion lambda:PublishVersion lambda:PutFunctionCodeSigningConfig lambda:PutFunctionConcurrency lambda:PutFunctionEventInvokeConfig lambda:PutFunctionRecursionConfig lambda:PutFunctionScalingConfig lambda:PutProvisionedConcurrencyConfig lambda:PutRuntimeManagementConfig lambda:RemoveLayerVersionPermission lambda:RemovePermission lambda:UpdateAlias lambda:UpdateCapacityProvider lambda:UpdateCodeSigningConfig

服务前缀	操作
	lambda:UpdateEventSourceMapping lambda:UpdateFunctionCode lambda:UpdateFunctionConfiguration lambda:UpdateFunctionEventInvokeConfig lambda:UpdateFunctionUrlConfig

服务前缀	操作
lex	lex:BatchCreateCustomVocabularyItem
	lex:BatchDeleteCustomVocabularyItem
	lex:BatchUpdateCustomVocabularyItem
	lex:BuildBotLocale
	lex:CreateBotAlias
	lex:CreateBotReplica
	lex:CreateBotVersion
	lex:CreateExport
	lex:CreateIntentVersion
	lex:CreateResourcePolicy
	lex:CreateSlotTypeVersion
	lex:CreateTestSetDiscrepancyReport
	lex:CreateUploadUrl
	lex:DeleteBot
	lex:DeleteBotChannelAssociation
	lex:DeleteBotReplica
	lex:DeleteExport
	lex:DeleteImport
	lex:DeleteIntentVersion
	lex:DeleteResourcePolicy
	lex:DeleteSlotTypeVersion

服务前缀	操作
	lex:DeleteTestSet
	lex:DeleteUtterances
	lex:DescribeBotAlias
	lex:DescribeBotRecommendation
	lex:DescribeBotReplica
	lex:DescribeBotResourceGeneration
	lex:DescribeBotVersion
	lex:DescribeCustomVocabularyMetadata
	lex:DescribeExport
	lex:DescribeImport
	lex:DescribeResourcePolicy
	lex:DescribeTestExecution
	lex:DescribeTestSet
	lex:DescribeTestSetDiscrepancyReport
	lex:DescribeTestSetGeneration
	lex:GenerateBotElement
	lex:GetBot
	lex:GetBotAlias
	lex:GetBotAliases
	lex:GetBotChannelAssociation
	lex:GetBotChannelAssociations

服务前缀	操作
	lex:GetBotVersions
	lex:GetBots
	lex:GetBuiltinIntent
	lex:GetBuiltinIntents
	lex:GetBuiltinSlotTypes
	lex:GetExport
	lex:GetImport
	lex:GetIntent
	lex:GetIntentVersions
	lex:GetIntents
	lex:GetMigration
	lex:GetMigrations
	lex:GetSlotType
	lex:GetSlotTypeVersions
	lex:GetSlotTypes
	lex:GetTestExecutionArtifactsUrl
	lex:GetUtterancesView
	lex:ListBotAliasReplicas
	lex:ListBotAliases
	lex:ListBotRecommendations
	lex:ListBotReplicas

服务前缀	操作
	lex:ListBotResourceGenerations
	lex:ListBotVersionReplicas
	lex:ListBotVersions
	lex:ListBots
	lex:ListBuiltInIntents
	lex:ListBuiltInSlotTypes
	lex:ListCustomVocabularyItems
	lex:ListExports
	lex:ListImports
	lex:ListIntentMetrics
	lex:ListIntentPaths
	lex:ListRecommendedIntents
	lex:ListSessionAnalyticsData
	lex:ListSessionMetrics
	lex:ListTestExecutionResultItems
	lex:ListTestExecutions
	lex:ListTestSets
	lex:PutBot
	lex:PutBotAlias
	lex:PutIntent
	lex:PutSlotType

服务前缀	操作
	lex:SearchAssociatedTranscripts lex:StartBotRecommendation lex:StartImport lex:StartMigration lex:StartTestExecution lex:StartTestSetGeneration lex:StopBotRecommendation lex:UpdateBotAlias lex:UpdateBotRecommendation lex:UpdateExport lex:UpdateResourcePolicy
license-manager-linux-subscriptions	license-manager-linux-subscriptions:DeregisterSubscriptionProvider license-manager-linux-subscriptions:GetRegisteredSubscriptionProvider license-manager-linux-subscriptions:GetServiceSettings license-manager-linux-subscriptions:ListLinuxSubscriptionInstances license-manager-linux-subscriptions:ListLinuxSubscriptions license-manager-linux-subscriptions:ListRegisteredSubscriptionProviders license-manager-linux-subscriptions:RegisterSubscriptionProvider license-manager-linux-subscriptions:UpdateServiceSettings

服务前缀	操作
lightsail	lightsail:AllocateStaticIp
	lightsail:AttachCertificateToDistribution
	lightsail:AttachDisk
	lightsail:AttachInstancesToLoadBalancer
	lightsail:AttachLoadBalancerTlsCertificate
	lightsail:AttachStaticIp
	lightsail:CloseInstancePublicPorts
	lightsail:CopySnapshot
	lightsail:CreateBucket
	lightsail:CreateBucketAccessKey
	lightsail:CreateCertificate
	lightsail:CreateCloudFormationStack
	lightsail:CreateContactMethod
	lightsail:CreateContainerService
	lightsail:CreateContainerServiceDeployment
	lightsail:CreateContainerServiceRegistryLogin
	lightsail:CreateDisk
	lightsail:CreateDiskFromSnapshot
	lightsail:CreateDiskSnapshot
	lightsail:CreateDistribution
	lightsail:CreateDomain

服务前缀	操作
	lightsail:CreateGUISessionAccessDetails
	lightsail:CreateInstanceSnapshot
	lightsail:CreateInstances
	lightsail:CreateInstancesFromSnapshot
	lightsail:CreateKeyPair
	lightsail:CreateLoadBalancer
	lightsail:CreateLoadBalancerTlsCertificate
	lightsail:CreateRelationalDatabase
	lightsail:CreateRelationalDatabaseFromSnapshot
	lightsail:CreateRelationalDatabaseSnapshot
	lightsail:DeleteAlarm
	lightsail:DeleteAutoSnapshot
	lightsail:DeleteBucket
	lightsail:DeleteBucketAccessKey
	lightsail:DeleteCertificate
	lightsail:DeleteContactMethod
	lightsail:DeleteContainerImage
	lightsail:DeleteContainerService
	lightsail:DeleteDisk
	lightsail:DeleteDiskSnapshot
	lightsail:DeleteDistribution

服务前缀	操作
	lightsail:DeleteDomain
	lightsail:DeleteDomainEntry
	lightsail:DeleteInstance
	lightsail:DeleteInstanceSnapshot
	lightsail:DeleteKeyPair
	lightsail:DeleteKnownHostKeys
	lightsail:DeleteLoadBalancer
	lightsail:DeleteLoadBalancerTlsCertificate
	lightsail:DeleteRelationalDatabase
	lightsail:DeleteRelationalDatabaseSnapshot
	lightsail:DetachCertificateFromDistribution
	lightsail:DetachDisk
	lightsail:DetachInstancesFromLoadBalancer
	lightsail:DetachStaticIp
	lightsail:DisableAddOn
	lightsail:DownloadDefaultKeyPair
	lightsail:EnableAddOn
	lightsail:ExportSnapshot
	lightsail:GetActiveNames
	lightsail:GetAlarms
	lightsail:GetAutoSnapshots

服务前缀	操作
	lightsail:GetBlueprints
	lightsail:GetBucketAccessKeys
	lightsail:GetBucketBundles
	lightsail:GetBucketMetricData
	lightsail:GetBuckets
	lightsail:GetBundles
	lightsail:GetCertificates
	lightsail:GetCloudFormationStackRecords
	lightsail:GetContactMethods
	lightsail:GetContainerAPIMetadata
	lightsail:GetContainerImages
	lightsail:GetContainerLog
	lightsail:GetContainerServiceDeployments
	lightsail:GetContainerServiceMetricData
	lightsail:GetContainerServicePowers
	lightsail:GetContainerServices
	lightsail:GetCostEstimate
	lightsail:GetDisk
	lightsail:GetDiskSnapshot
	lightsail:GetDiskSnapshots
	lightsail:GetDisks

服务前缀	操作
	lightsail:GetDistributionBundles
	lightsail:GetDistributionLatestCacheReset
	lightsail:GetDistributionMetricData
	lightsail:GetDistributions
	lightsail:GetDomain
	lightsail:GetExportSnapshotRecords
	lightsail:GetInstance
	lightsail:GetInstanceMetricData
	lightsail:GetInstancePortStates
	lightsail:GetInstanceSnapshot
	lightsail:GetInstanceSnapshots
	lightsail:GetInstanceState
	lightsail:GetInstances
	lightsail:GetKeyPair
	lightsail:GetKeyPairs
	lightsail:GetLoadBalancer
	lightsail:GetLoadBalancerMetricData
	lightsail:GetLoadBalancerTlsCertificates
	lightsail:GetLoadBalancerTlsPolicies
	lightsail:GetLoadBalancers
	lightsail:GetOperation

服务前缀	操作
	lightsail:GetOperations
	lightsail:GetOperationsForResource
	lightsail:GetRegions
	lightsail:GetRelationalDatabase
	lightsail:GetRelationalDatabaseBlueprints
	lightsail:GetRelationalDatabaseBundles
	lightsail:GetRelationalDatabaseEvents
	lightsail:GetRelationalDatabaseLogEvents
	lightsail:GetRelationalDatabaseLogStreams
	lightsail:GetRelationalDatabaseMasterUserPassword
	lightsail:GetRelationalDatabaseMetricData
	lightsail:GetRelationalDatabaseParameters
	lightsail:GetRelationalDatabaseSnapshot
	lightsail:GetRelationalDatabaseSnapshots
	lightsail:GetRelationalDatabases
	lightsail:GetSetupHistory
	lightsail:GetStaticIp
	lightsail:GetStaticIps
	lightsail:ImportKeyPair
	lightsail:IsVpcPeered
	lightsail:OpenInstancePublicPorts

服务前缀	操作
	lightsail:PeerVpc
	lightsail:PutAlarm
	lightsail:PutInstancePublicPorts
	lightsail:RebootInstance
	lightsail:RebootRelationalDatabase
	lightsail:RegisterContainerImage
	lightsail:releaseStatiCip
	lightsail:ResetDistributionCache
	lightsail:SendContactMethodVerification
	lightsail:SetIpAddressType
	lightsail:SetResourceAccessForBucket
	lightsail:SetupInstanceHttps
	lightsail:StartGUISession
	lightsail:StartInstance
	lightsail:StartRelationalDatabase
	lightsail:StopGUISession
	lightsail:StopInstance
	lightsail:StopRelationalDatabase
	lightsail:TestAlarm
	lightsail:UnpeerVpc
	lightsail:UpdateBucket

服务前缀	操作
	lightsail:UpdateBucketBundle lightsail:UpdateContainerService lightsail:UpdateDistribution lightsail:UpdateDistributionBundle lightsail:UpdateDomainEntry lightsail:UpdateInstanceMetadataOptions lightsail:UpdateLoadBalancerAttribute lightsail:UpdateRelationalDatabase lightsail:UpdateRelationalDatabaseParameters

服务前缀	操作
logs	logs:AssociateKmsKey logs:AssociateSourceToS3TableIntegration logs:CancelExportTask logs:CancelImportTask logs:CreateDelivery logs:CreateExportTask logs:CreateLogAnomalyDetector logs:CreateLogGroup logs:CreateLogStream logs:DeleteDataProtectionPolicy logs:DeleteDelivery logs:DeleteDeliveryDestination logs:DeleteDeliveryDestinationPolicy logs:DeleteDeliverySource logs:DeleteDestination logs:DeleteIndexPolicy logs:DeleteIntegration logs:DeleteLogAnomalyDetector logs:DeleteLogGroup logs:DeleteLogStream logs:DeleteMetricFilter

服务前缀	操作
	logs:DeleteQueryDefinition logs:DeleteResourcePolicy logs:DeleteRetentionPolicy logs:DeleteScheduledQuery logs:DeleteSubscriptionFilter logs:DeleteTransformer logs:DescribeAccountPolicies logs:DescribeConfigurationTemplates logs:DescribeDeliveries logs:DescribeDeliveryDestinations logs:DescribeDeliverySources logs:DescribeDestinations logs:DescribeExportTasks logs:DescribeFieldIndexes logs:DescribeImportTaskBatches logs:DescribeImportTasks logs:DescribeIndexPolicies logs:DescribeLogGroups logs:DescribeLogStreams logs:DescribeMetricFilters logs:DescribeQueries

服务前缀	操作
	logs:DescribeQueryDefinitions logs:DescribeResourcePolicies logs:DescribeSubscriptionFilters logs:DisassociateKmsKey logs:DisassociateSourceFromS3TableIntegration logs:GetDataProtectionPolicy logs:GetDelivery logs:GetDeliveryDestination logs:GetDeliveryDestinationPolicy logs:GetDeliverySource logs:GetIntegration logs:GetLogAnomalyDetector logs:GetLogFields logs:GetLogGroupFields logs:GetLogRecord logs:GetQueryResults logs:GetScheduledQuery logs:GetScheduledQueryHistory logs:GetTransformer logs:ListAnomalies logs:ListIntegrations

服务前缀	操作
	logs:ListLogAnomalyDetectors logs:ListLogGroupsForQuery logs:ListScheduledQueries logs:ListSourcesForS3TableIntegration logs:PutDataProtectionPolicy logs:PutDeliveryDestination logs:PutDeliveryDestinationPolicy logs:PutDeliverySource logs:PutDestination logs:PutDestinationPolicy logs:PutIndexPolicy logs:PutIntegration logs:PutLogGroupDeletionProtection logs:PutMetricFilter logs:PutQueryDefinition logs:PutResourcePolicy logs:PutRetentionPolicy logs:PutSubscriptionFilter logs:PutTransformer logs:StartLiveTail logs:StartQuery

服务前缀	操作
	logs:StopQuery logs:TestMetricFilter logs:TestTransformer logs:UpdateAnomaly logs:UpdateDeliveryConfiguration logs:UpdateLogAnomalyDetector

服务前缀	操作
lookoutequipment	lookoutequipment:CreateDataset lookoutequipment:CreateInferenceScheduler lookoutequipment:CreateLabel lookoutequipment:CreateLabelGroup lookoutequipment:CreateModel lookoutequipment:DeleteDataset lookoutequipment:DeleteInferenceScheduler lookoutequipment:DeleteLabel lookoutequipment:DeleteLabelGroup lookoutequipment:DeleteModel lookoutequipment:DeleteResourcePolicy lookoutequipment:DeleteRetrainingScheduler lookoutequipment:DescribeDataIngestionJob lookoutequipment:DescribeDataset lookoutequipment:DescribeInferenceScheduler lookoutequipment:DescribeLabelGroup lookoutequipment:DescribeModel lookoutequipment:DescribeModelVersion lookoutequipment:DescribeResourcePolicy lookoutequipment:DescribeRetrainingScheduler lookoutequipment:Describelabel

服务前缀	操作
	lookoutequipment:ImportDataset
	lookoutequipment:ImportModelVersion
	lookoutequipment:ListDataIngestionJobs
	lookoutequipment:ListDatasets
	lookoutequipment:ListInferenceEvents
	lookoutequipment:ListInferenceExecutions
	lookoutequipment:ListInferenceSchedulers
	lookoutequipment:ListLabelGroups
	lookoutequipment:ListLabels
	lookoutequipment:ListModelVersions
	lookoutequipment:ListModels
	lookoutequipment:ListRetrainingSchedulers
	lookoutequipment:ListSensorStatistics
	lookoutequipment:PutResourcePolicy
	lookoutequipment:StartDataIngestionJob
	lookoutequipment:StartInferenceScheduler
	lookoutequipment:StartRetrainingScheduler
	lookoutequipment:StopInferenceScheduler
	lookoutequipment:StopRetrainingScheduler
	lookoutequipment:UpdateActiveModelVersion
	lookoutequipment:UpdateInferenceScheduler

服务前缀	操作
	lookoutequipment:UpdateLabelGroup lookoutequipment:UpdateModel lookoutequipment:UpdateRetrainingScheduler

服务前缀	操作
lookoutmetrics	lookoutmetrics:ActivateAnomalyDetector lookoutmetrics:BackTestAnomalyDetector lookoutmetrics:CreateAlert lookoutmetrics:CreateAnomalyDetector lookoutmetrics:CreateMetricSet lookoutmetrics:DeactivateAnomalyDetector lookoutmetrics:DeleteAlert lookoutmetrics:DeleteAnomalyDetector lookoutmetrics:DescribeAlert lookoutmetrics:DescribeAnomalyDetectionExecutions lookoutmetrics:DescribeAnomalyDetector lookoutmetrics:DescribeMetricSet lookoutmetrics:DetectMetricSetConfig lookoutmetrics:GetAnomalyGroup lookoutmetrics:GetDataQualityMetrics lookoutmetrics:GetFeedback lookoutmetrics:GetSampleData lookoutmetrics:ListAlerts lookoutmetrics:ListAnomalyDetectors lookoutmetrics:ListAnomalyGroupRelatedMetrics lookoutmetrics:ListAnomalyGroupSummaries

服务前缀	操作
	lookoutmetrics:ListAnomalyGroupTimeSeries lookoutmetrics:ListMetricSets lookoutmetrics:PutFeedback lookoutmetrics:UpdateAlert lookoutmetrics:UpdateAnomalyDetector lookoutmetrics:UpdateMetricSet

服务前缀	操作
lookoutvision	lookoutvision:CreateDataset lookoutvision:CreateModel lookoutvision:CreateProject lookoutvision:DeleteDataset lookoutvision:DeleteModel lookoutvision:DeleteProject lookoutvision:DescribeDataset lookoutvision:DescribeModel lookoutvision:DescribeModelPackagingJob lookoutvision:DescribeProject lookoutvision:DetectAnomalies lookoutvision:ListDatasetEntries lookoutvision:ListModelPackagingJobs lookoutvision:ListModels lookoutvision:ListProjects lookoutvision:StartModel lookoutvision:StartModelPackagingJob lookoutvision:StopModel lookoutvision:UpdateDatasetEntries

服务前缀	操作
m2	m2:CancelBatchJobExecution m2:CreateApplication m2:CreateDataSetExportTask m2:CreateDataSetImportTask m2:CreateDeployment m2:CreateEnvironment m2:DeleteApplication m2:DeleteApplicationFromEnvironment m2:DeleteEnvironment m2:GetApplication m2:GetApplicationVersion m2:GetBatchJobExecution m2:GetDataSetDetails m2:GetDataSetExportTask m2:GetDataSetImportTask m2:GetDeployment m2:GetEnvironment m2:GetSignedBluinsightsUrl m2:ListApplicationVersions m2:ListApplications m2:ListBatchJobDefinitions

服务前缀	操作
	m2:ListBatchJobExecutions m2:ListBatchJobRestartPoints m2:ListDataSetExportHistory m2:ListDataSetImportHistory m2:ListDataSets m2:ListDeployments m2:ListEngineVersions m2:ListEnvironments m2:StartApplication m2:StartBatchJob m2:StopApplication m2:UpdateApplication m2:UpdateEnvironment

服务前缀	操作
managedblockchain	managedblockchain:CreateAccessor
	managedblockchain:CreateMember
	managedblockchain:CreateNetwork
	managedblockchain:CreateNode
	managedblockchain:CreateProposal
	managedblockchain:DeleteAccessor
	managedblockchain:DeleteMember
	managedblockchain:DeleteNode
	managedblockchain:GetAccessor
	managedblockchain:GetMember
	managedblockchain:GetNetwork
	managedblockchain:GetNode
	managedblockchain:GetProposal
	managedblockchain:InvokeRpcPolygonMainnet
	managedblockchain:InvokeRpcPolygonMumbaiTestnet
	managedblockchain:ListAccessors
	managedblockchain:ListInvitations
	managedblockchain:ListMembers
	managedblockchain:ListNetworks
	managedblockchain:ListNodes
	managedblockchain:ListProposalVotes

服务前缀	操作
	managedblockchain:ListProposals
	managedblockchain:RejectInvitation
	managedblockchain:UpdateMember
	managedblockchain:UpdateNode
	managedblockchain:VoteOnProposal

服务前缀	操作
mediaconnect	mediaconnect:AddBridgeOutputs mediaconnect:AddBridgeSources mediaconnect:AddFlowMediaStreams mediaconnect:AddFlowOutputs mediaconnect:AddFlowSources mediaconnect:AddFlowVpcInterfaces mediaconnect:CreateBridge mediaconnect:CreateFlow mediaconnect:CreateGateway mediaconnect:DeleteBridge mediaconnect:DeleteFlow mediaconnect:DeleteGateway mediaconnect:DeleteRouterInput mediaconnect:DeleteRouterNetworkInterface mediaconnect:DeleteRouterOutput mediaconnect:DeregisterGatewayInstance mediaconnect:DescribeBridge mediaconnect:DescribeFlow mediaconnect:DescribeFlowSourceMetadata mediaconnect:DescribeFlowSourceThumbnail mediaconnect:DescribeGateway

服务前缀	操作
	mediaconnect:DescribeGatewayInstance mediaconnect:DescribeOffering mediaconnect:DescribeReservation mediaconnect:GetRouterInput mediaconnect:GetRouterInputSourceMetadata mediaconnect:GetRouterInputThumbnail mediaconnect:GetRouterNetworkInterface mediaconnect:GetRouterOutput mediaconnect:GrantFlowEntitlements mediaconnect:ListBridges mediaconnect:ListEntitlements mediaconnect:ListFlows mediaconnect:ListGatewayInstances mediaconnect:ListGateways mediaconnect:ListOfferings mediaconnect:ListReservations mediaconnect:ListRouterInputs mediaconnect:ListRouterNetworkInterfaces mediaconnect:ListRouterOutputs mediaconnect:PurchaseOffering mediaconnect:RemoveBridgeOutput

服务前缀	操作
	mediaconnect:RemoveBridgeSource mediaconnect:RemoveFlowMediaStream mediaconnect:RemoveFlowOutput mediaconnect:RemoveFlowSource mediaconnect:RemoveFlowVpcInterface mediaconnect:RestartRouterInput mediaconnect:RestartRouterOutput mediaconnect:RevokeFlowEntitlement mediaconnect:StartFlow mediaconnect:StartRouterInput mediaconnect:StartRouterOutput mediaconnect:StopFlow mediaconnect:StopRouterInput mediaconnect:StopRouterOutput mediaconnect:TakeRouterInput mediaconnect:UpdateBridge mediaconnect:UpdateBridgeOutput mediaconnect:UpdateBridgeSource mediaconnect:UpdateBridgeState mediaconnect:UpdateFlow mediaconnect:UpdateFlowEntitlement

服务前缀	操作
	mediaconnect:UpdateFlowMediaStream mediaconnect:UpdateGatewayInstance

服务前缀	操作
mediaconvert	mediaconvert:AssociateCertificate
	mediaconvert:CancelJob
	mediaconvert:CreateJob
	mediaconvert:CreateJobTemplate
	mediaconvert:CreatePreset
	mediaconvert:CreateQueue
	mediaconvert:CreateResourceShare
	mediaconvert:DeleteJobTemplate
	mediaconvert:DeletePolicy
	mediaconvert:DeletePreset
	mediaconvert:DeleteQueue
	mediaconvert:DescribeEndpoints
	mediaconvert:DisassociateCertificate
	mediaconvert:GetJob
	mediaconvert:GetJobTemplate
	mediaconvert:GetPolicy
	mediaconvert:GetPreset
	mediaconvert:GetQueue
	mediaconvert:ListJobTemplates
	mediaconvert:ListJobs
	mediaconvert:ListPresets

服务前缀	操作
	mediaconvert:ListQueues
	mediaconvert:ListVersions
	mediaconvert:Probe
	mediaconvert:PutPolicy
	mediaconvert:SearchJobs
	mediaconvert:UpdateJobTemplate
	mediaconvert:UpdatePreset
	mediaconvert:UpdateQueue

服务前缀	操作
medialive	medialive:AcceptInputDeviceTransfer medialive:BatchDelete medialive:BatchStart medialive:BatchStop medialive:BatchUpdateSchedule medialive:CancelInputDeviceTransfer medialive:ClaimDevice medialive:CreateChannel medialive:CreateChannelPlacementGroup medialive:CreateCloudWatchAlarmTemplate medialive:CreateCloudWatchAlarmTemplateGroup medialive:CreateCluster medialive:CreateEventBridgeRuleTemplate medialive:CreateEventBridgeRuleTemplateGroup medialive:CreateInput medialive:CreateInputSecurityGroup medialive:CreateMultiplex medialive:CreateMultiplexProgram medialive:CreateNetwork medialive:CreateNode medialive:CreateNodeRegistrationScript

服务前缀	操作
	medialive:CreatePartnerInput medialive:CreateSdiSource medialive:CreateSignalMap medialive:DeleteChannel medialive:DeleteChannelPlacementGroup medialive:DeleteCloudWatchAlarmTemplate medialive:DeleteCloudWatchAlarmTemplateGroup medialive:DeleteCluster medialive:DeleteEventBridgeRuleTemplate medialive:DeleteEventBridgeRuleTemplateGroup medialive:DeleteInput medialive:DeleteInputSecurityGroup medialive:DeleteMultiplex medialive:DeleteMultiplexProgram medialive:DeleteNetwork medialive:DeleteNode medialive:DeleteReservation medialive:DeleteSchedule medialive:DeleteSdiSource medialive:DeleteSignalMap medialive:DescribeAccountConfiguration

服务前缀	操作
	medialive:DescribeChannel medialive:DescribeChannelPlacementGroup medialive:DescribeCluster medialive:DescribeInput medialive:DescribeInputDevice medialive:DescribeInputDeviceThumbnail medialive:DescribeInputSecurityGroup medialive:DescribeMultiplex medialive:DescribeMultiplexProgram medialive:DescribeNetwork medialive:DescribeNode medialive:DescribeOffering medialive:DescribeReservation medialive:DescribeSchedule medialive:DescribeSdiSource medialive:DescribeThumbnails medialive:GetCloudWatchAlarmTemplate medialive:GetCloudWatchAlarmTemplateGroup medialive:GetEventBridgeRuleTemplate medialive:GetEventBridgeRuleTemplateGroup medialive:GetSignalMap

服务前缀	操作
	medialive:ListAlerts medialive:ListChannelPlacementGroups medialive:ListChannels medialive:ListCloudWatchAlarmTemplateGroups medialive:ListCloudWatchAlarmTemplates medialive:ListClusterAlerts medialive:ListClusters medialive:ListEventBridgeRuleTemplateGroups medialive:ListEventBridgeRuleTemplates medialive:ListInputDeviceTransfers medialive:ListInputDevices medialive:ListInputSecurityGroups medialive:ListInputs medialive:ListMultiplexAlerts medialive:ListMultiplexPrograms medialive:ListMultiplexes medialive:ListNetworks medialive:ListNodes medialive:ListOfferings medialive:ListReservations medialive:ListSdiSources

服务前缀	操作
	medialive:ListSignalMaps medialive:ListVersions medialive:PurchaseOffering medialive:RebootInputDevice medialive:RejectInputDeviceTransfer medialive:RestartChannelPipelines medialive:StartChannel medialive:StartDeleteMonitorDeployment medialive:StartInputDevice medialive:StartInputDeviceMaintenanceWindow medialive:StartMonitorDeployment medialive:StartMultiplex medialive:StartUpdateSignalMap medialive:StopChannel medialive:StopInputDevice medialive:StopMultiplex medialive:TransferInputDevice medialive:UpdateAccountConfiguration medialive:UpdateChannel medialive:UpdateChannelClass medialive:UpdateChannelPlacementGroup

服务前缀	操作
	medialive:UpdateCloudWatchAlarmTemplate medialive:UpdateCloudWatchAlarmTemplateGroup medialive:UpdateCluster medialive:UpdateEventBridgeRuleTemplate medialive:UpdateEventBridgeRuleTemplateGroup medialive:UpdateInput medialive:UpdateInputDevice medialive:UpdateInputSecurityGroup medialive:UpdateMultiplex medialive:UpdateMultiplexProgram medialive:UpdateNetwork medialive:UpdateNode medialive:UpdateNodeState medialive:UpdateReservation medialive:UpdateSdiSource

服务前缀	操作
mediastore	mediastore:CreateContainer
	mediastore:DeleteContainer
	mediastore:DeleteContainerPolicy
	mediastore:DeleteCorsPolicy
	mediastore:DeleteLifecyclePolicy
	mediastore:DeleteMetricPolicy
	mediastore:DescribeContainer
	mediastore:GetContainerPolicy
	mediastore:GetCorsPolicy
	mediastore:GetLifecyclePolicy
	mediastore:GetMetricPolicy
	mediastore:ListContainers
	mediastore:PutContainerPolicy
	mediastore:PutCorsPolicy
	mediastore:PutLifecyclePolicy
	mediastore:PutMetricPolicy
	mediastore:StartAccessLogging
	mediastore:StopAccessLogging

服务前缀	操作
mediatailor	mediatailor:ConfigureLogsForPlaybackConfiguration mediatailor:CreateChannel mediatailor:CreateLiveSource mediatailor:CreatePrefetchSchedule mediatailor:CreateProgram mediatailor:CreateSourceLocation mediatailor:CreateVodSource mediatailor:DeleteChannel mediatailor:DeleteChannelPolicy mediatailor:DeleteLiveSource mediatailor:DeletePlaybackConfiguration mediatailor:DeletePrefetchSchedule mediatailor:DeleteProgram mediatailor:DeleteSourceLocation mediatailor:DeleteVodSource mediatailor:DescribeChannel mediatailor:DescribeLiveSource mediatailor:DescribeProgram mediatailor:DescribeSourceLocation mediatailor:DescribeVodSource mediatailor:GetChannelPolicy

服务前缀	操作
	mediatailor:GetChannelSchedule mediatailor:GetPlaybackConfiguration mediatailor:GetPrefetchSchedule mediatailor:ListAlerts mediatailor:ListChannels mediatailor:ListLiveSources mediatailor:ListPlaybackConfigurations mediatailor:ListPrefetchSchedules mediatailor:ListSourceLocations mediatailor:ListVodSources mediatailor:PutChannelPolicy mediatailor:PutPlaybackConfiguration mediatailor:StartChannel mediatailor:StopChannel mediatailor:UpdateChannel mediatailor:UpdateLiveSource mediatailor:UpdateProgram mediatailor:UpdateSourceLocation mediatailor:UpdateVodSource

服务前缀	操作
memorydb	memorydb:BatchUpdateCluster
	memorydb:CopySnapshot
	memorydb:CreateAcl
	memorydb:CreateCluster
	memorydb:CreateMultiRegionCluster
	memorydb:CreateParameterGroup
	memorydb:CreateSnapshot
	memorydb:CreateSubnetGroup
	memorydb:CreateUser
	memorydb:DeleteAcl
	memorydb:DeleteCluster
	memorydb:DeleteMultiRegionCluster
	memorydb:DeleteParameterGroup
	memorydb:DeleteSnapshot
	memorydb:DeleteSubnetGroup
	memorydb:DeleteUser
	memorydb:DescribeAcls
	memorydb:DescribeClusters
	memorydb:DescribeEngineVersions
	memorydb:DescribeEvents
	memorydb:DescribeMultiRegionClusters

服务前缀	操作
	memorydb:DescribeMultiRegionParameterGroups
	memorydb:DescribeMultiRegionParameters
	memorydb:DescribeParameterGroups
	memorydb:DescribeParameters
	memorydb:DescribeReservedNodes
	memorydb:DescribeReservedNodesOfferings
	memorydb:DescribeServiceUpdates
	memorydb:DescribeSnapshots
	memorydb:DescribeSubnetGroups
	memorydb:DescribeUsers
	memorydb:FailoverShard
	memorydb:ListAllowedMultiRegionClusterUpdates
	memorydb:ListAllowedNodeTypeUpdates
	memorydb:PurchaseReservedNodesOffering
	memorydb:ResetParameterGroup
	memorydb:UpdateAcl
	memorydb:UpdateCluster
	memorydb:UpdateMultiRegionCluster
	memorydb:UpdateParameterGroup
	memorydb:UpdateSubnetGroup
	memorydb:UpdateUser

服务前缀	操作
mgh	mgh:AssociateCreatedArtifact mgh:AssociateDiscoveredResource mgh:AssociateSourceResource mgh:CreateHomeRegionControl mgh:CreateProgressUpdateStream mgh:DeleteHomeRegionControl mgh:DeleteProgressUpdateStream mgh:DescribeApplicationState mgh:DescribeHomeRegionControls mgh:DescribeMigrationTask mgh:DisassociateCreatedArtifact mgh:DisassociateDiscoveredResource mgh:DisassociateSourceResource mgh:GetHomeRegion mgh:ImportMigrationTask mgh:ListApplicationStates mgh:ListCreatedArtifacts mgh:ListDiscoveredResources mgh:ListMigrationTaskUpdates mgh:ListMigrationTasks mgh:ListProgressUpdateStreams

服务前缀	操作
	mgh:ListSourceResources mgh:NotifyApplicationState mgh:NotifyMigrationTaskState mgh:PutResourceAttributes

服务前缀	操作
mgn	mgn:ArchiveApplication
	mgn:ArchiveWave
	mgn:AssociateApplications
	mgn:AssociateSourceServers
	mgn:ChangeServerLifeCycleState
	mgn:CreateApplication
	mgn:CreateConnector
	mgn:CreateLaunchConfigurationTemplate
	mgn:CreateReplicationConfigurationTemplate
	mgn:CreateWave
	mgn:DeleteApplication
	mgn:DeleteConnector
	mgn:DeleteJob
	mgn:DeleteLaunchConfigurationTemplate
	mgn:DeleteReplicationConfigurationTemplate
	mgn:DeleteSourceServer
	mgn:DeleteVcenterClient
	mgn:DeleteWave
	mgn:DescribeJobLogItems
	mgn:DescribeJobs
	mgn:DescribeLaunchConfigurationTemplates

服务前缀	操作
	mgn:DescribeReplicationConfigurationTemplates
	mgn:DescribeVcenterClients
	mgn:DisassociateApplications
	mgn:DisassociateSourceServers
	mgn:DisconnectFromService
	mgn:FinalizeCutover
	mgn:GetReplicationConfiguration
	mgn:InitializeService
	mgn:ListConnectors
	mgn:ListExportErrors
	mgn:ListExports
	mgn:ListImportErrors
	mgn:ListImports
	mgn:ListManagedAccounts
	mgn:ListSourceServerActions
	mgn:ListTemplateActions
	mgn:MarkAsArchived
	mgn:PauseReplication
	mgn:PutSourceServerAction
	mgn:PutTemplateAction
	mgn:RemoveSourceServerAction

服务前缀	操作
	mgn:RemoveTemplateAction
	mgn:ResumeReplication
	mgn:RetryDataReplication
	mgn:StartCutover
	mgn:StartExport
	mgn:StartImport
	mgn:StartReplication
	mgn:StartTest
	mgn:StopReplication
	mgn:TerminateTargetInstances
	mgn:UnarchiveApplication
	mgn:UnarchiveWave
	mgn:UpdateApplication
	mgn:UpdateConnector
	mgn:UpdateLaunchConfigurationTemplate
	mgn:UpdateReplicationConfiguration
	mgn:UpdateReplicationConfigurationTemplate
	mgn:UpdateSourceServer
	mgn:UpdateSourceServerReplicationType
	mgn:UpdateWave

服务前缀	操作
migrationhub-strategy	migrationhub-strategy:GetAntiPattern migrationhub-strategy:GetApplicationComponentDetails migrationhub-strategy:GetApplicationComponentStrategies migrationhub-strategy:GetAssessment migrationhub-strategy:GetImportFileTask migrationhub-strategy:GetLatestAssessmentId migrationhub-strategy:GetMessage migrationhub-strategy:GetPortfolioPreferences migrationhub-strategy:GetPortfolioSummary migrationhub-strategy:GetRecommendationReportDetails migrationhub-strategy:GetServerDetails migrationhub-strategy:GetServerStrategies migrationhub-strategy:ListAnalyzableServers migrationhub-strategy:ListAntiPatterns migrationhub-strategy:ListApplicationComponents migrationhub-strategy:ListCollectors migrationhub-strategy:ListImportFileTask migrationhub-strategy:ListJarArtifacts migrationhub-strategy:ListServers migrationhub-strategy:PutLogData migrationhub-strategy:PutMetricData

服务前缀	操作
	migrationhub-strategy:PutPortfolioPreferences migrationhub-strategy:RegisterCollector migrationhub-strategy:SendMessage migrationhub-strategy:StartAssessment migrationhub-strategy:StartImportFileTask migrationhub-strategy:StartRecommendationReportGeneration migrationhub-strategy:StopAssessment migrationhub-strategy:UpdateApplicationComponentConfig migrationhub-strategy:UpdateCollectorConfiguration migrationhub-strategy:UpdateServerConfig

服务前缀	操作
mobiletargeting	mobiletargeting:CreateApp
	mobiletargeting:CreateCampaign
	mobiletargeting:CreateEmailTemplate
	mobiletargeting:CreateExportJob
	mobiletargeting:CreateImportJob
	mobiletargeting:CreateInAppTemplate
	mobiletargeting:CreateJourney
	mobiletargeting:CreatePushTemplate
	mobiletargeting:CreateRecommenderConfiguration
	mobiletargeting:CreateSegment
	mobiletargeting:CreateSmsTemplate
	mobiletargeting:CreateVoiceTemplate
	mobiletargeting:DeleteAdmChannel
	mobiletargeting:DeleteApnsChannel
	mobiletargeting:DeleteApnsSandboxChannel
	mobiletargeting:DeleteApnsVoipChannel
	mobiletargeting:DeleteApnsVoipSandboxChannel
	mobiletargeting:DeleteApp
	mobiletargeting:DeleteBaiduChannel
	mobiletargeting:DeleteCampaign
	mobiletargeting:DeleteEmailChannel

服务前缀	操作
	mobiletargeting:DeleteEmailTemplate
	mobiletargeting:DeleteEndpoint
	mobiletargeting:DeleteEventStream
	mobiletargeting:DeleteGcmChannel
	mobiletargeting:DeleteInAppTemplate
	mobiletargeting:DeleteJourney
	mobiletargeting:DeletePushTemplate
	mobiletargeting:DeleteRecommenderConfiguration
	mobiletargeting:DeleteSegment
	mobiletargeting:DeleteSmsChannel
	mobiletargeting:DeleteSmsTemplate
	mobiletargeting:DeleteUserEndpoints
	mobiletargeting:DeleteVoiceChannel
	mobiletargeting:DeleteVoiceTemplate
	mobiletargeting:GetAdmChannel
	mobiletargeting:GetApnsChannel
	mobiletargeting:GetApnsSandboxChannel
	mobiletargeting:GetApnsVoipChannel
	mobiletargeting:GetApnsVoipSandboxChannel
	mobiletargeting:GetApp
	mobiletargeting:GetApplicationDateRangeKpi

服务前缀	操作
	mobiletargeting:GetApplicationSettings
	mobiletargeting:GetApps
	mobiletargeting:GetBaiduChannel
	mobiletargeting:GetCampaign
	mobiletargeting:GetCampaignActivities
	mobiletargeting:GetCampaignDateRangeKpi
	mobiletargeting:GetCampaignVersion
	mobiletargeting:GetCampaignVersions
	mobiletargeting:GetCampaigns
	mobiletargeting:GetChannels
	mobiletargeting:GetEmailChannel
	mobiletargeting:GetEmailTemplate
	mobiletargeting:GetEndpoint
	mobiletargeting:GetEventStream
	mobiletargeting:GetExportJob
	mobiletargeting:GetExportJobs
	mobiletargeting:GetGcmChannel
	mobiletargeting:GetImportJob
	mobiletargeting:GetImportJobs
	mobiletargeting:GetInAppMessages
	mobiletargeting:GetInAppTemplate

服务前缀	操作
	mobiletargeting:GetJourney
	mobiletargeting:GetJourneyDateRangeKpi
	mobiletargeting:GetJourneyExecutionActivityMetrics
	mobiletargeting:GetJourneyExecutionMetrics
	mobiletargeting:GetJourneyRunExecutionActivityMetrics
	mobiletargeting:GetJourneyRunExecutionMetrics
	mobiletargeting:GetJourneyRuns
	mobiletargeting:GetPushTemplate
	mobiletargeting:GetRecommenderConfiguration
	mobiletargeting:GetRecommenderConfigurations
	mobiletargeting:GetSegment
	mobiletargeting:GetSegmentExportJobs
	mobiletargeting:GetSegmentImportJobs
	mobiletargeting:GetSegmentVersion
	mobiletargeting:GetSegmentVersions
	mobiletargeting:GetSegments
	mobiletargeting:GetSmsChannel
	mobiletargeting:GetSmsTemplate
	mobiletargeting:GetUserEndpoints
	mobiletargeting:GetVoiceChannel
	mobiletargeting:GetVoiceTemplate

服务前缀	操作
	mobiletargeting:ListJourneys
	mobiletargeting:ListTemplateVersions
	mobiletargeting:ListTemplates
	mobiletargeting:PhoneNumberValidate
	mobiletargeting:PutEventStream
	mobiletargeting:RemoveAttributes
	mobiletargeting:UpdateAdmChannel
	mobiletargeting:UpdateApnsChannel
	mobiletargeting:UpdateApnsSandboxChannel
	mobiletargeting:UpdateApnsVoipChannel
	mobiletargeting:UpdateApnsVoipSandboxChannel
	mobiletargeting:UpdateApplicationSettings
	mobiletargeting:UpdateBaiduChannel
	mobiletargeting:UpdateCampaign
	mobiletargeting:UpdateEmailChannel
	mobiletargeting:UpdateEmailTemplate
	mobiletargeting:UpdateEndpoint
	mobiletargeting:UpdateEndpointsBatch
	mobiletargeting:UpdateGcmChannel
	mobiletargeting:UpdateInAppTemplate
	mobiletargeting:UpdateJourney

服务前缀	操作
	mobiletargeting:UpdateJourneyState
	mobiletargeting:UpdatePushTemplate
	mobiletargeting:UpdateRecommenderConfiguration
	mobiletargeting:UpdateSegment
	mobiletargeting:UpdateSmsChannel
	mobiletargeting:UpdateSmsTemplate
	mobiletargeting:UpdateTemplateActiveVersion
	mobiletargeting:UpdateVoiceChannel
	mobiletargeting:UpdateVoiceTemplate
	mobiletargeting:VerifyOTPMessage

服务前缀	操作
mq	mq:CreateBroker mq:CreateConfiguration mq:CreateUser mq:DeleteBroker mq:DeleteConfiguration mq:DeleteUser mq:DescribeBroker mq:DescribeBrokerEngineTypes mq:DescribeBrokerInstanceOptions mq:DescribeConfiguration mq:DescribeConfigurationRevision mq:DescribeUser mq:ListBrokers mq:ListConfigurationRevisions mq:ListConfigurations mq:ListUsers mq:Promote mq:RebootBroker mq:UpdateBroker mq:UpdateConfiguration mq:UpdateUser

服务前缀	操作
networkmanager	networkmanager:AcceptAttachment
	networkmanager:AssociateConnectPeer
	networkmanager:AssociateCustomerGateway
	networkmanager:AssociateLink
	networkmanager:AssociateTransitGatewayConnectPeer
	networkmanager:CreateConnectAttachment
	networkmanager:CreateConnectPeer
	networkmanager:CreateConnection
	networkmanager:CreateCoreNetwork
	networkmanager:CreateDevice
	networkmanager:CreateDirectConnectGatewayAttachment
	networkmanager:CreateGlobalNetwork
	networkmanager:CreateLink
	networkmanager:CreateSite
	networkmanager:CreateSiteToSiteVpnAttachment
	networkmanager:CreateTransitGatewayPeering
	networkmanager:CreateTransitGatewayRouteTableAttachment
	networkmanager:CreateVpcAttachment
	networkmanager:DeleteAttachment
	networkmanager:DeleteConnectPeer
	networkmanager:DeleteConnection

服务前缀	操作
	networkmanager:DeleteCoreNetwork
	networkmanager:DeleteCoreNetworkPolicyVersion
	networkmanager:DeleteDevice
	networkmanager:DeleteGlobalNetwork
	networkmanager:DeleteLink
	networkmanager:DeletePeering
	networkmanager:DeleteResourcePolicy
	networkmanager:DeleteSite
	networkmanager:DeregisterTransitGateway
	networkmanager:DescribeGlobalNetworks
	networkmanager:DisassociateConnectPeer
	networkmanager:DisassociateCustomerGateway
	networkmanager:DisassociateLink
	networkmanager:DisassociateTransitGatewayConnectPeer
	networkmanager:ExecuteCoreNetworkChangeSet
	networkmanager:GetConnectAttachment
	networkmanager:GetConnectPeer
	networkmanager:GetConnectPeerAssociations
	networkmanager:GetConnections
	networkmanager:GetCoreNetwork
	networkmanager:GetCoreNetworkChangeEvents

服务前缀	操作
	networkmanager:GetCoreNetworkChangeSet
	networkmanager:GetCoreNetworkPolicy
	networkmanager:GetCustomerGatewayAssociations
	networkmanager:GetDevices
	networkmanager:GetLinkAssociations
	networkmanager:GetLinks
	networkmanager:GetNetworkResourceCounts
	networkmanager:GetNetworkResourceRelationships
	networkmanager:GetNetworkResources
	networkmanager:GetNetworkRoutes
	networkmanager:GetNetworkTelemetry
	networkmanager:GetResourcePolicy
	networkmanager:GetRouteAnalysis
	networkmanager:GetSiteToSiteVpnAttachment
	networkmanager:GetSites
	networkmanager:GetTransitGatewayConnectPeerAssociations
	networkmanager:GetTransitGatewayPeering
	networkmanager:GetTransitGatewayRegistrations
	networkmanager:GetTransitGatewayRouteTableAttachment
	networkmanager:GetVpcAttachment
	networkmanager:ListAttachmentRoutingPolicyAssociations

服务前缀	操作
	networkmanager:ListAttachments
	networkmanager:ListConnectPeers
	networkmanager:ListCoreNetworkPolicyVersions
	networkmanager:ListCoreNetworkPrefixListAssociations
	networkmanager:ListCoreNetworkRoutingInformation
	networkmanager:ListCoreNetworks
	networkmanager:ListOrganizationServiceAccessStatus
	networkmanager:ListPeerings
	networkmanager:PutAttachmentRoutingPolicyLabel
	networkmanager:PutCoreNetworkPolicy
	networkmanager:PutResourcePolicy
	networkmanager:RegisterTransitGateway
	networkmanager:RejectAttachment
	networkmanager:RemoveAttachmentRoutingPolicyLabel
	networkmanager:RestoreCoreNetworkPolicyVersion
	networkmanager:StartOrganizationServiceAccessUpdate
	networkmanager:StartRouteAnalysis
	networkmanager:UpdateConnection
	networkmanager:UpdateCoreNetwork
	networkmanager:UpdateDevice
	networkmanager:UpdateDirectConnectGatewayAttachment

服务前缀	操作
	networkmanager:UpdateGlobalNetwork networkmanager:UpdateLink networkmanager:UpdateNetworkResourceMetadata networkmanager:UpdateSite networkmanager:UpdateVpcAttachment

服务前缀	操作
nimble	nimble:AcceptEulas nimble:CreateLaunchProfile nimble:CreateStreamingImage nimble:CreateStreamingSession nimble:CreateStreamingSessionStream nimble:CreateStudio nimble:CreateStudioComponent nimble:DeleteLaunchProfile nimble:DeleteLaunchProfileMember nimble:DeleteStreamingImage nimble:DeleteStreamingSession nimble:DeleteStudio nimble:DeleteStudioComponent nimble:DeleteStudioMember nimble:GetEula nimble:GetLaunchProfileDetails nimble:GetStreamingImage nimble:GetStreamingSession nimble:GetStreamingSessionBackup nimble:GetStreamingSessionStream nimble:GetStudio

服务前缀	操作
	nimble:GetStudioComponent nimble:GetStudioMember nimble:ListEulas nimble:ListLaunchProfileMembers nimble:ListLaunchProfiles nimble:ListStreamingImages nimble:ListStreamingSessionBackups nimble:ListStreamingSessions nimble:ListStudioComponents nimble:ListStudioMembers nimble:ListStudios nimble:PutLaunchProfileMembers nimble:PutStudioMembers nimble:StartStreamingSession nimble:StartStudioSSOConfigurationRepair nimble:StopStreamingSession nimble:UpdateLaunchProfile nimble:UpdateLaunchProfileMember nimble:UpdateStreamingImage nimble:UpdateStudio nimble:UpdateStudioComponent

服务前缀	操作
omics	omics:AbortMultipartReadSetUpload
	omics:AcceptShare
	omics:BatchDeleteReadSet
	omics:CancelAnnotationImportJob
	omics:CancelRun
	omics:CancelVariantImportJob
	omics:CompleteMultipartReadSetUpload
	omics:CreateAnnotationStore
	omics:CreateAnnotationStoreVersion
	omics:CreateMultipartReadSetUpload
	omics:CreateReferenceStore
	omics:CreateRunGroup
	omics:CreateSequenceStore
	omics:CreateShare
	omics:CreateVariantStore
	omics:CreateWorkflow
	omics:CreateWorkflowVersion
	omics:DeleteAnnotationStore
	omics:DeleteAnnotationStoreVersions
	omics:DeleteReference
	omics:DeleteReferenceStore

服务前缀	操作
	omics:DeleteRun
	omics:DeleteRunGroup
	omics:DeleteSequenceStore
	omics:DeleteShare
	omics:DeleteVariantStore
	omics:DeleteWorkflow
	omics:DeleteWorkflowVersion
	omics:GetAnnotationImportJob
	omics:GetAnnotationStore
	omics:GetAnnotationStoreVersion
	omics:GetReadSet
	omics:GetReadSetActivationJob
	omics:GetReadSetExportJob
	omics:GetReadSetImportJob
	omics:GetReadSetMetadata
	omics:GetReference
	omics:GetReferenceImportJob
	omics:GetReferenceMetadata
	omics:GetReferenceStore
	omics:GetRun
	omics:GetRunGroup

服务前缀	操作
	omics:GetRunTask
	omics:GetSequenceStore
	omics:GetShare
	omics:GetVariantImportJob
	omics:GetVariantStore
	omics:GetWorkflow
	omics:GetWorkflowVersion
	omics:ListAnnotationImportJobs
	omics:ListAnnotationStoreVersions
	omics:ListAnnotationStores
	omics:ListMultipartReadSetUploads
	omics:ListReadSetActivationJobs
	omics:ListReadSetExportJobs
	omics:ListReadSetImportJobs
	omics:ListReadSetUploadParts
	omics:ListReadSets
	omics:ListReferenceImportJobs
	omics:ListReferenceStores
	omics:ListReferences
	omics:ListRunGroups
	omics:ListRunTasks

服务前缀	操作
	omics:ListRuns
	omics:ListSequenceStores
	omics:ListShares
	omics:ListVariantImportJobs
	omics:ListVariantStores
	omics:ListWorkflowVersions
	omics:ListWorkflows
	omics:StartAnnotationImportJob
	omics:StartReadSetActivationJob
	omics:StartReadSetExportJob
	omics:StartReadSetImportJob
	omics:StartReferenceImportJob
	omics:StartRun
	omics:StartVariantImportJob
	omics:UpdateAnnotationStore
	omics:UpdateAnnotationStoreVersion
	omics:UpdateRunGroup
	omics:UpdateVariantStore
	omics:UpdateWorkflow
	omics:UpdateWorkflowVersion
	omics:UploadReadSetPart

服务前缀	操作
opsworks	opsworks:AssignInstance opsworks:AssignVolume opsworks:AssociateElasticIp opsworks:AttachElasticLoadBalancer opsworks:CloneStack opsworks:CreateApp opsworks:CreateDeployment opsworks:CreateInstance opsworks:CreateLayer opsworks:CreateStack opsworks:CreateUserProfile opsworks:DeleteApp opsworks:DeleteInstance opsworks:DeleteLayer opsworks:DeleteStack opsworks:DeleteUserProfile opsworks:DeregisterEcsCluster opsworks:DeregisterElasticIp opsworks:DeregisterInstance opsworks:DeregisterRdsDbInstance opsworks:DeregisterVolume

服务前缀	操作
	opsworks:DescribeAgentVersions opsworks:DescribeApps opsworks:DescribeCommands opsworks:DescribeDeployments opsworks:DescribeEcsClusters opsworks:DescribeElasticIps opsworks:DescribeElasticLoadBalancers opsworks:DescribeInstances opsworks:DescribeLayers opsworks:DescribeLoadBasedAutoScaling opsworks:DescribeMyUserProfile opsworks:DescribeOperatingSystems opsworks:DescribePermissions opsworks:DescribeRaidArrays opsworks:DescribeRdsDbInstances opsworks:DescribeServiceErrors opsworks:DescribeStackProvisioningParameters opsworks:DescribeStackSummary opsworks:DescribeStacks opsworks:DescribeTimeBasedAutoScaling opsworks:DescribeUserProfiles

服务前缀	操作
	opsworks:DescribeVolumes opsworks:DetachElasticLoadBalancer opsworks:DisassociateElasticIp opsworks:GetHostnameSuggestion opsworks:GrantAccess opsworks:RebootInstance opsworks:RegisterEcsCluster opsworks:RegisterElasticIp opsworks:RegisterInstance opsworks:RegisterRdsDbInstance opsworks:RegisterVolume opsworks:SetLoadBasedAutoScaling opsworks:SetPermission opsworks:SetTimeBasedAutoScaling opsworks:StartInstance opsworks:StartStack opsworks:StopInstance opsworks:StopStack opsworks:UnassignInstance opsworks:UnassignVolume opsworks:UpdateApp

服务前缀	操作
	opsworks:UpdateElasticIp opsworks:UpdateInstance opsworks:UpdateLayer opsworks:UpdateMyUserProfile opsworks:UpdateRdsDbInstance opsworks:UpdateStack opsworks:UpdateUserProfile opsworks:UpdateVolume

服务前缀	操作
opsworks-cm	opsworks-cm:AssociateNode
	opsworks-cm:CreateBackup
	opsworks-cm:CreateServer
	opsworks-cm:DeleteBackup
	opsworks-cm:DeleteServer
	opsworks-cm:DescribeAccountAttributes
	opsworks-cm:DescribeBackups
	opsworks-cm:DescribeEvents
	opsworks-cm:DescribeNodeAssociationStatus
	opsworks-cm:DescribeServers
	opsworks-cm:DisassociateNode
	opsworks-cm:ExportServerEngineAttribute
	opsworks-cm:RestoreServer
	opsworks-cm:StartMaintenance
	opsworks-cm:UpdateServer
	opsworks-cm:UpdateServerEngineAttributes

服务前缀	操作
组织	organizations:AcceptHandshake organizations:AttachPolicy organizations:CancelHandshake organizations:CloseAccount organizations:CreateAccount organizations:CreateGovCloudAccount organizations:CreateOrganization organizations:CreateOrganizationalUnit organizations:CreatePolicy organizations:DeclineHandshake organizations:DeleteOrganization organizations:DeleteOrganizationalUnit organizations:DeletePolicy organizations:DeleteResourcePolicy organizations:DeregisterDelegatedAdministrator organizations:DescribeAccount organizations:DescribeCreateAccountStatus organizations:DescribeEffectivePolicy organizations:DescribeHandshake organizations:DescribeOrganization organizations:DescribeOrganizationalUnit

服务前缀	操作
	organizations:DescribePolicy organizations:DescribeResourcePolicy organizations:DescribeResponsibilityTransfer organizations:DetachPolicy organizations:DisableAWSServiceAccess organizations:DisablePolicyType organizations:EnableAWSServiceAccess organizations:EnableAllFeatures organizations:EnablePolicyType organizations:InviteAccountToOrganization organizations:LeaveOrganization organizations:ListAWSServiceAccessForOrganization organizations:ListAccounts organizations:ListAccountsForParent organizations:ListAccountsWithInvalidEffectivePolicy organizations:ListChildren organizations:ListCreateAccountStatus organizations:ListDelegatedAdministrators organizations:ListDelegatedServicesForAccount organizations:ListHandshakesForAccount organizations:ListHandshakesForOrganization

服务前缀	操作
	organizations:ListInboundResponsibilityTransfers organizations:ListOrganizationalUnitsForParent organizations:ListOutboundResponsibilityTransfers organizations:ListParents organizations:ListPolicies organizations:ListPoliciesForTarget organizations:ListRoots organizations:ListTargetsForPolicy organizations:MoveAccount organizations:PutResourcePolicy organizations:RegisterDelegatedAdministrator organizations:RemoveAccountFromOrganization organizations:TerminateResponsibilityTransfer organizations:UpdateOrganizationalUnit organizations:UpdatePolicy organizations:UpdateResponsibilityTransfer

服务前缀	操作
outposts	outposts:CancelCapacityTask
	outposts:CancelOrder
	outposts:CreateOrder
	outposts:CreateOutpost
	outposts:CreatePrivateConnectivityConfig
	outposts:CreateSite
	outposts:DeleteOutpost
	outposts:DeleteSite
	outposts:GetCapacityTask
	outposts:GetCatalogItem
	outposts:GetConnection
	outposts:GetOrder
	outposts:GetOutpost
	outposts:GetOutpostBillingInformation
	outposts:GetOutpostInstanceTypes
	outposts:GetOutpostSupportedInstanceTypes
	outposts:GetPrivateConnectivityConfig
	outposts:GetSite
	outposts:GetSiteAddress
	outposts:ListAssetInstances
	outposts:ListAssets

服务前缀	操作
	outposts:ListBlockingInstancesForCapacityTask
	outposts:ListCapacityTasks
	outposts:ListCatalogItems
	outposts:ListOrders
	outposts:ListOutposts
	outposts:ListSites
	outposts:StartCapacityTask
	outposts:StartConnection
	outposts:UpdateOutpost
	outposts:UpdateSite
	outposts:UpdateSiteAddress
	outposts:UpdateSiteRackPhysicalProperties

服务前缀	操作
panorama	panorama:CreateApplicationInstance panorama:CreateJobForDevices panorama:CreateNodeFromTemplateJob panorama:CreatePackage panorama:CreatePackageImportJob panorama:DeleteDevice panorama:DeletePackage panorama:DeregisterPackageVersion panorama:DescribeApplicationInstance panorama:DescribeApplicationInstanceDetails panorama:DescribeDevice panorama:DescribeDeviceJob panorama:DescribeNode panorama:DescribeNodeFromTemplateJob panorama:DescribePackage panorama:DescribePackageImportJob panorama:DescribePackageVersion panorama:ListApplicationInstanceDependencies panorama:ListApplicationInstanceNodeInstances panorama:ListApplicationInstances panorama:ListDevices

服务前缀	操作
	panorama:ListDevicesJobs panorama:ListNodeFromTemplateJobs panorama:ListNodes panorama:ListPackageImportJobs panorama:ListPackages panorama:ProvisionDevice panorama:RegisterPackageVersion panorama:RemoveApplicationInstance panorama:SignalApplicationInstanceNodeInstances panorama:UpdateDeviceMetadata
pi	pi:CreatePerformanceAnalysisReport pi:DeletePerformanceAnalysisReport pi:DescribeDimensionKeys pi:GetDimensionKeyDetails pi:GetPerformanceAnalysisReport pi:GetResourceMetadata pi:GetResourceMetrics pi:ListAvailableResourceDimensions pi:ListAvailableResourceMetrics pi:ListPerformanceAnalysisReports

服务前缀	操作
pipes	pipes:CreatePipe pipes:DeletePipe pipes:DescribePipe pipes:ListPipes pipes:StartPipe pipes:StopPipe pipes:UpdatePipe
polly	polly:DeleteLexicon polly:DescribeVoices polly:GetLexicon polly:GetSpeechSynthesisTask polly:ListLexicons polly:ListSpeechSynthesisTasks polly:PutLexicon polly:StartSpeechSynthesisTask polly:SynthesizeSpeech

服务前缀	操作
配置文件	profile:AddProfileKey profile:BatchGetCalculatedAttributeForProfile profile:BatchGetProfile profile:CreateCalculatedAttributeDefinition profile:CreateDomain profile:CreateEventStream profile:CreateProfile profile:CreateRecommender profile:CreateSegmentDefinition profile:CreateSegmentEstimate profile:CreateSegmentSnapshot profile:CreateUploadJob profile:DeleteCalculatedAttributeDefinition profile:DeleteDomain profile:DeleteDomainObjectType profile:DeleteEventStream profile:DeleteIntegration profile:DeleteProfile profile:DeleteProfileKey profile:DeleteProfileObject profile:DeleteProfileObjectType

服务前缀	操作
	profile:DeleteRecommender profile:DeleteSegmentDefinition profile:DeleteWorkflow profile:DetectProfileObjectType profile:GetAutoMergingPreview profile:GetCalculatedAttributeDefinition profile:GetCalculatedAttributeForProfile profile:GetDomain profile:GetDomainObjectType profile:GetEventStream profile:GetIdentityResolutionJob profile:GetIntegration profile:GetMatches profile:GetObjectTypeAttributeStatistics profile:GetProfileObjectType profile:GetProfileObjectTypeTemplate profile:GetProfileRecommendations profile:GetRecommender profile:GetSegmentDefinition profile:GetSegmentEstimate profile:GetSegmentMembership

服务前缀	操作
	profile:GetSegmentSnapshot profile:GetSimilarProfiles profile:GetUploadJob profile:GetUploadJobPath profile:GetWorkflow profile:GetWorkflowSteps profile:ListAccountIntegrations profile:ListCalculatedAttributeDefinitions profile:ListCalculatedAttributesForProfile profile:ListDomainLayouts profile:ListDomainObjectTypes profile:ListDomains profile:ListEventStreams profile:ListIdentityResolutionJobs profile:ListIntegrations profile:ListObjectTypeAttributeValues profile:ListObjectTypeAttributes profile:ListProfileAttributeValues profile:ListProfileObjectTypeTemplates profile:ListProfileObjectTypes profile:ListProfileObjects

服务前缀	操作
	profile:ListRecommenderRecipes profile:ListRecommenders profile:ListRuleBasedMatches profile:ListSegmentDefinitions profile:ListUploadJobs profile:ListWorkflows profile:MergeProfiles profile:PutDomainObjectType profile:PutIntegration profile:PutProfileObject profile:PutProfileObjectType profile:SearchProfiles profile:StartRecommender profile:StartUploadJob profile:StopRecommender profile:StopUploadJob profile:UpdateCalculatedAttributeDefinition profile:UpdateDomain profile:UpdateProfile profile:UpdateRecommender

服务前缀	操作
qldb	qldb:CancelJournalKinesisStream qldb:CreateLedger qldb:DeleteLedger qldb:DescribeJournalKinesisStream qldb:DescribeJournalS3Export qldb:DescribeLedger qldb:ExportJournalToS3 qldb:GetBlock qldb:GetDigest qldb:GetRevision qldb:ListJournalKinesisStreamsForLedger qldb:ListJournalS3Exports qldb:ListJournalS3ExportsForLedger qldb:ListLedgers qldb:StreamJournalToKinesis qldb:UpdateLedger qldb:UpdateLedgerPermissionsMode

服务前缀	操作
ram	ram:AcceptResourceShareInvitation ram:AssociateResourceShare ram:AssociateResourceSharePermission ram:CreatePermission ram:CreatePermissionVersion ram:CreateResourceShare ram:DeletePermission ram:DeletePermissionVersion ram:DeleteResourceShare ram:DisassociateResourceShare ram:DisassociateResourceSharePermission ram:EnableSharingWithAwsOrganization ram:GetPermission ram:GetResourcePolicies ram:GetResourceShareAssociations ram:GetResourceShareInvitations ram:GetResourceShares ram:ListPendingInvitationResources ram:ListPermissionAssociations ram:ListPermissionVersions ram:ListPermissions

服务前缀	操作
	ram:ListPrincipals ram:ListReplacePermissionAssociationsWork ram:ListResourceSharePermissions ram:ListResourceTypes ram:ListResources ram:PromotePermissionCreatedFromPolicy ram:PromoteResourceShareCreatedFromPolicy ram:RejectResourceShareInvitation ram:ReplacePermissionAssociations ram:SetDefaultPermissionVersion ram:UpdateResourceShare
rbin	rbin:CreateRule rbin:DeleteRule rbin:GetRule rbin:ListRules rbin:LockRule rbin:UnlockRule rbin:UpdateRule

服务前缀	操作
rds	rds:AddRoleToDBCluster rds:AddRoleToDBInstance rds:AddSourceIdentifierToSubscription rds:ApplyPendingMaintenanceAction rds:AuthorizeDBSecurityGroupIngress rds:BacktrackDBCluster rds:CancelExportTask rds:CopyDBClusterParameterGroup rds:CopyDBClusterSnapshot rds:CopyDBParameterGroup rds:CopyDBSnapshot rds:CopyOptionGroup rds:CreateCustomDBEngineVersion rds:CreateDBClusterParameterGroup rds:CreateDBParameterGroup rds:CreateDBProxy rds:CreateDBProxyEndpoint rds:CreateDBSecurityGroup rds:CreateDBSubnetGroup rds:CreateEventSubscription rds:CreateGlobalCluster

服务前缀	操作
	rds:CreateOptionGroup rds:DeleteBlueGreenDeployment rds:DeleteDBClusterAutomatedBackup rds:DeleteDBClusterParameterGroup rds:DeleteDBClusterSnapshot rds:DeleteDBInstanceAutomatedBackup rds:DeleteDBParameterGroup rds:DeleteDBProxy rds:DeleteDBProxyEndpoint rds:DeleteDBSecurityGroup rds:DeleteDBSnapshot rds:DeleteDBSubnetGroup rds:DeleteEventSubscription rds:DeleteGlobalCluster rds:DeleteOptionGroup rds:DeregisterDBProxyTargets rds:DescribeAccountAttributes rds:DescribeBlueGreenDeployments rds:DescribeCertificates rds:DescribeDBClusterAutomatedBackups rds:DescribeDBClusterBacktracks

服务前缀	操作
	rds:DescribeDBClusterEndpoints rds:DescribeDBClusterParameterGroups rds:DescribeDBClusterParameters rds:DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshots rds:DescribeDBClusters rds:DescribeDBEngineVersions rds:DescribeDBInstanceAutomatedBackups rds:DescribeDBInstances rds:DescribeDBLogFiles rds:DescribeDBMajorEngineVersions rds:DescribeDBParameterGroups rds:DescribeDBParameters rds:DescribeDBProxies rds:DescribeDBProxyEndpoints rds:DescribeDBProxyTargetGroups rds:DescribeDBProxyTargets rds:DescribeDBRecommendations rds:DescribeDBSecurityGroups rds:DescribeDBSnapshotAttributes rds:DescribeDBSnapshotTenantDatabases

服务前缀	操作
	rds:DescribeDBSnapshots rds:DescribeDBSubnetGroups rds:DescribeEngineDefaultClusterParameters rds:DescribeEngineDefaultParameters rds:DescribeEventCategories rds:DescribeEventSubscriptions rds:DescribeEvents rds:DescribeExportTasks rds:DescribeGlobalClusters rds:DescribeIntegrations rds:DescribeOptionGroupOptions rds:DescribeOptionGroups rds:DescribeOrderableDBInstanceOptions rds:DescribePendingMaintenanceActions rds:DescribeReservedDBInstances rds:DescribeReservedDBInstancesOfferings rds:DescribeSourceRegions rds:DescribeTenantDatabases rds:DescribeValidDBInstanceModifications rds:DownloadCompleteDBLogFile rds:DownloadDBLogFilePortion

服务前缀	操作
	rds:FailoverDBCluster rds:FailoverGlobalCluster rds:ModifyActivityStream rds:ModifyCertificates rds:ModifyCurrentDBClusterCapacity rds:ModifyDBClusterEndpoint rds:ModifyDBClusterParameterGroup rds:ModifyDBClusterSnapshotAttribute rds:ModifyDBParameterGroup rds:ModifyDBProxy rds:ModifyDBProxyEndpoint rds:ModifyDBProxyTargetGroup rds:ModifyDBRecommendation rds:ModifyDBSnapshot rds:ModifyDBSnapshotAttribute rds:ModifyDBSubnetGroup rds:ModifyEventSubscription rds:ModifyGlobalCluster rds:ModifyOptionGroup rds:ModifyTenantDatabase rds:PurchaseReservedDBInstancesOffering

服务前缀	操作
	rds:RebootDBCluster rds:RegisterDBProxyTargets rds:RemoveFromGlobalCluster rds:RemoveRoleFromDBCluster rds:RemoveRoleFromDBInstance rds:RemoveSourceIdentifierFromSubscription rds:ResetDBClusterParameterGroup rds:ResetDBParameterGroup rds:RestoreDBClusterFromS3 rds:RestoreDBClusterFromSnapshot rds:RestoreDBClusterToPointInTime rds:RestoreDBInstanceFromDBSnapshot rds:RestoreDBInstanceFromS3 rds:RestoreDBInstanceToPointInTime rds:RevokeDBSecurityGroupIngress rds:StartActivityStream rds:StartDBCluster rds:StartDBInstance rds:StartDBInstanceAutomatedBackupsReplication rds:StartExportTask rds:StopActivityStream

服务前缀	操作
	rds:StopDBCluster rds:StopDBInstance rds:StopDBInstanceAutomatedBackupsReplication rds:SwitchoverBlueGreenDeployment rds:SwitchoverGlobalCluster rds:SwitchoverReadReplica

服务前缀	操作
redshift	redshift:AcceptReservedNodeExchange
	redshift:AddPartner
	redshift:AssociateDataShareConsumer
	redshift:AuthorizeClusterSecurityGroupIngress
	redshift:AuthorizeDataShare
	redshift:AuthorizeEndpointAccess
	redshift:AuthorizeSnapshotAccess
	redshift:BatchDeleteClusterSnapshots
	redshift:BatchModifyClusterSnapshots
	redshift:CancelResize
	redshift:CopyClusterSnapshot
	redshift:CreateAuthenticationProfile
	redshift:CreateCluster
	redshift:CreateClusterParameterGroup
	redshift:CreateClusterSecurityGroup
	redshift:CreateClusterSnapshot
	redshift:CreateClusterSubnetGroup
	redshift:CreateCustomDomainAssociation
	redshift:CreateEndpointAccess
	redshift:CreateEventSubscription
	redshift:CreateHsmClientCertificate

服务前缀	操作
	redshift:CreateHsmConfiguration
	redshift:CreateIntegration
	redshift:CreateRedshiftIdcApplication
	redshift:CreateScheduledAction
	redshift:CreateSnapshotCopyGrant
	redshift:CreateSnapshotSchedule
	redshift:CreateUsageLimit
	redshift:DeauthorizeDataShare
	redshift:DeleteAuthenticationProfile
	redshift:DeleteCluster
	redshift:DeleteClusterParameterGroup
	redshift:DeleteClusterSecurityGroup
	redshift:DeleteClusterSnapshot
	redshift:DeleteClusterSubnetGroup
	redshift:DeleteCustomDomainAssociation
	redshift:DeleteEndpointAccess
	redshift:DeleteEventSubscription
	redshift:DeleteHsmClientCertificate
	redshift:DeleteHsmConfiguration
	redshift:DeletePartner
	redshift:DeleteRedshiftIdcApplication

服务前缀	操作
	redshift:DeleteResourcePolicy
	redshift:DeleteScheduledAction
	redshift:DeleteSnapshotCopyGrant
	redshift:DeleteSnapshotSchedule
	redshift:DeleteUsageLimit
	redshift:DeregisterNamespace
	redshift:DescribeAccountAttributes
	redshift:DescribeAuthenticationProfiles
	redshift:DescribeClusterDbRevisions
	redshift:DescribeClusterParameterGroups
	redshift:DescribeClusterParameters
	redshift:DescribeClusterSecurityGroups
	redshift:DescribeClusterSnapshots
	redshift:DescribeClusterSubnetGroups
	redshift:DescribeClusterTracks
	redshift:DescribeClusterVersions
	redshift:DescribeClusters
	redshift:DescribeCustomDomainAssociations
	redshift:DescribeDataShares
	redshift:DescribeDataSharesForConsumer
	redshift:DescribeDataSharesForProducer

服务前缀	操作
	redshift:DescribeDefaultClusterParameters
	redshift:DescribeEndpointAccess
	redshift:DescribeEndpointAuthorization
	redshift:DescribeEventCategories
	redshift:DescribeEventSubscriptions
	redshift:DescribeEvents
	redshift:DescribeHsmClientCertificates
	redshift:DescribeHsmConfigurations
	redshift:DescribeInboundIntegrations
	redshift:DescribeIntegrations
	redshift:DescribeLoggingStatus
	redshift:DescribeNodeConfigurationOptions
	redshift:DescribeOrderableClusterOptions
	redshift:DescribePartners
	redshift:DescribeRedshiftIdcApplications
	redshift:DescribeReservedNodeExchangeStatus
	redshift:DescribeReservedNodeOfferings
	redshift:DescribeReservedNodes
	redshift:DescribeResize
	redshift:DescribeScheduledActions
	redshift:DescribeSnapshotCopyGrants

服务前缀	操作
	redshift:DescribeSnapshotSchedules
	redshift:DescribeStorage
	redshift:DescribeTableRestoreStatus
	redshift:DescribeUsageLimits
	redshift:DisableLogging
	redshift:DisableSnapshotCopy
	redshift:DisassociateDataShareConsumer
	redshift:EnableLogging
	redshift:EnableSnapshotCopy
	redshift:FailoverPrimaryCompute
	redshift:GetClusterCredentials
	redshift:GetClusterCredentialsWithIAM
	redshift:GetIdentityCenterAuthToken
	redshift:GetReservedNodeExchangeConfigurationOptions
	redshift:GetReservedNodeExchangeOfferings
	redshift:GetResourcePolicy
	redshift:ListRecommendations
	redshift:ModifyAquaConfiguration
	redshift:ModifyAuthenticationProfile
	redshift:ModifyCluster
	redshift:ModifyClusterDbRevision

服务前缀	操作
	redshift:ModifyClusterIamRoles
	redshift:ModifyClusterMaintenance
	redshift:ModifyClusterParameterGroup
	redshift:ModifyClusterSnapshot
	redshift:ModifyClusterSnapshotSchedule
	redshift:ModifyClusterSubnetGroup
	redshift:ModifyCustomDomainAssociation
	redshift:ModifyEndpointAccess
	redshift:ModifyEventSubscription
	redshift:ModifyRedshiftIdcApplication
	redshift:ModifyScheduledAction
	redshift:ModifySnapshotCopyRetentionPeriod
	redshift:ModifySnapshotSchedule
	redshift:ModifyUsageLimit
	redshift:PauseCluster
	redshift:PurchaseReservedNodeOffering
	redshift:PutResourcePolicy
	redshift:RebootCluster
	redshift:RegisterNamespace
	redshift:RejectDataShare
	redshift:ResetClusterParameterGroup

服务前缀	操作
	redshift:ResizeCluster redshift:RestoreFromClusterSnapshot redshift:RestoreTableFromClusterSnapshot redshift:ResumeCluster redshift:RevokeClusterSecurityGroupIngress redshift:RevokeEndpointAccess redshift:RevokeSnapshotAccess redshift:RotateEncryptionKey redshift:UpdatePartnerStatus
redshift-data	redshift-data:BatchExecuteStatement redshift-data:CancelStatement redshift-data:DescribeStatement redshift-data:DescribeTable redshift-data:ExecuteStatement redshift-data:GetStatementResult redshift-data:ListDatabases redshift-data:ListSchemas redshift-data:ListStatements redshift-data:ListTables

服务前缀	操作
refactor-spaces	refactor-spaces:CreateApplication
	refactor-spaces:CreateEnvironment
	refactor-spaces:CreateRoute
	refactor-spaces:CreateService
	refactor-spaces:DeleteApplication
	refactor-spaces:DeleteEnvironment
	refactor-spaces:DeleteResourcePolicy
	refactor-spaces:DeleteRoute
	refactor-spaces:DeleteService
	refactor-spaces:GetApplication
	refactor-spaces:GetEnvironment
	refactor-spaces:GetResourcePolicy
	refactor-spaces:GetRoute
	refactor-spaces:GetService
	refactor-spaces:ListApplications
	refactor-spaces:ListEnvironmentVpcs
	refactor-spaces:ListEnvironments
	refactor-spaces:ListRoutes
	refactor-spaces:ListServices
	refactor-spaces:PutResourcePolicy
	refactor-spaces:UpdateRoute

服务前缀	操作
rekognition	rekognition:AssociateFaces rekognition:CompareFaces rekognition:CopyProjectVersion rekognition:CreateCollection rekognition:CreateDataset rekognition:CreateFaceLivenessSession rekognition:CreateProject rekognition:CreateProjectVersion rekognition:CreateStreamProcessor rekognition:CreateUser rekognition:DeleteCollection rekognition:DeleteDataset rekognition:DeleteFaces rekognition:DeleteProject rekognition:DeleteProjectPolicy rekognition:DeleteProjectVersion rekognition:DeleteStreamProcessor rekognition:DeleteUser rekognition:DescribeCollection rekognition:DescribeDataset rekognition:DescribeProjectVersions

服务前缀	操作
	rekognition:DescribeProjects rekognition:DescribeStreamProcessor rekognition:DetectCustomLabels rekognition:DetectFaces rekognition:DetectLabels rekognition:DetectModerationLabels rekognition:DetectProtectiveEquipment rekognition:DetectText rekognition:DisassociateFaces rekognition:DistributeDatasetEntries rekognition:GetCelebrityInfo rekognition:GetCelebrityRecognition rekognition:GetContentModeration rekognition:GetFaceDetection rekognition:GetFaceLivenessSessionResults rekognition:GetFaceSearch rekognition:GetLabelDetection rekognition:GetMediaAnalysisJob rekognition:GetPersonTracking rekognition:GetSegmentDetection rekognition:GetTextDetection

服务前缀	操作
	rekognition:IndexFaces rekognition:ListCollections rekognition:ListDatasetEntries rekognition:ListDatasetLabels rekognition:ListFaces rekognition:ListMediaAnalysisJobs rekognition:ListProjectPolicies rekognition:ListStreamProcessors rekognition:ListUsers rekognition:PutProjectPolicy rekognition:RecognizeCelebrities rekognition:SearchFaces rekognition:SearchFacesByImage rekognition:SearchUsers rekognition:SearchUsersByImage rekognition:StartCelebrityRecognition rekognition:StartContentModeration rekognition:StartFaceDetection rekognition:StartFaceLivenessSession rekognition:StartFaceSearch rekognition:StartLabelDetection

服务前缀	操作
	rekognition:StartMediaAnalysisJob rekognition:StartPersonTracking rekognition:StartProjectVersion rekognition:StartSegmentDetection rekognition:StartStreamProcessor rekognition:StartTextDetection rekognition:StopProjectVersion rekognition:StopStreamProcessor rekognition:UpdateDatasetEntries rekognition:UpdateStreamProcessor

服务前缀	操作
resiliencehub	resiliencehub:AcceptResourceGroupingRecommendations
	resiliencehub:AddDraftAppVersionResourceMappings
	resiliencehub:BatchUpdateRecommendationStatus
	resiliencehub:CreateApp
	resiliencehub:CreateAppVersionAppComponent
	resiliencehub:CreateAppVersionResource
	resiliencehub:CreateRecommendationTemplate
	resiliencehub:CreateResiliencyPolicy
	resiliencehub:DeleteApp
	resiliencehub:DeleteAppAssessment
	resiliencehub:DeleteAppInputSource
	resiliencehub:DeleteAppVersionAppComponent
	resiliencehub:DeleteAppVersionResource
	resiliencehub:DeleteRecommendationTemplate
	resiliencehub:DeleteResiliencyPolicy
	resiliencehub:DescribeApp
	resiliencehub:DescribeAppAssessment
	resiliencehub:DescribeAppVersion
	resiliencehub:DescribeAppVersionAppComponent
	resiliencehub:DescribeAppVersionResource
	resiliencehub:DescribeAppVersionResourcesResolutionStatus

服务前缀	操作
	resiliencehub:DescribeAppVersionTemplate
	resiliencehub:DescribeDraftAppVersionResourcesImportStatus
	resiliencehub:DescribeMetricsExport
	resiliencehub:DescribeResiliencyPolicy
	resiliencehub:DescribeResourceGroupingRecommendationTask
	resiliencehub:ImportResourcesToDraftAppVersion
	resiliencehub:ListAlarmRecommendations
	resiliencehub:ListAppAssessmentComplianceDrifts
	resiliencehub:ListAppAssessmentResourceDrifts
	resiliencehub:ListAppAssessments
	resiliencehub:ListAppComponentCompliances
	resiliencehub:ListAppComponentRecommendations
	resiliencehub:ListAppInputSources
	resiliencehub:ListAppVersionAppComponents
	resiliencehub:ListAppVersionResourceMappings
	resiliencehub:ListAppVersionResources
	resiliencehub:ListAppVersions
	resiliencehub:ListApps
	resiliencehub:ListMetrics
	resiliencehub:ListRecommendationTemplates
	resiliencehub:ListResiliencyPolicies

服务前缀	操作
	resiliencehub:ListResourceGroupingRecommendations
	resiliencehub:ListSopRecommendations
	resiliencehub:ListSuggestedResiliencyPolicies
	resiliencehub:ListTestRecommendations
	resiliencehub:ListUnsupportedAppVersionResources
	resiliencehub:PublishAppVersion
	resiliencehub:PutDraftAppVersionTemplate
	resiliencehub:RejectResourceGroupingRecommendations
	resiliencehub:RemoveDraftAppVersionResourceMappings
	resiliencehub:ResolveAppVersionResources
	resiliencehub:StartAppAssessment
	resiliencehub:StartResourceGroupingRecommendationTask
	resiliencehub:UpdateApp
	resiliencehub:UpdateAppVersion
	resiliencehub:UpdateAppVersionAppComponent
	resiliencehub:UpdateAppVersionResource
	resiliencehub:UpdateResiliencyPolicy

服务前缀	操作
resource-explorer-2	resource-explorer-2:AssociateDefaultView resource-explorer-2:BatchGetView resource-explorer-2:CreateIndex resource-explorer-2:CreateResourceExplorerSetup resource-explorer-2:CreateView resource-explorer-2:DeleteIndex resource-explorer-2:DeleteResourceExplorerSetup resource-explorer-2:DeleteView resource-explorer-2:DisassociateDefaultView resource-explorer-2:GetAccountLevelServiceConfiguration resource-explorer-2:GetDefaultView resource-explorer-2:GetIndex resource-explorer-2:GetManagedView resource-explorer-2:GetResourceExplorerSetup resource-explorer-2:GetServiceIndex resource-explorer-2:GetServiceView resource-explorer-2:ListIndexes resource-explorer-2:ListIndexesForMembers resource-explorer-2:ListManagedViews resource-explorer-2:ListServiceIndexes resource-explorer-2:ListServiceViews

服务前缀	操作
	resource-explorer-2:ListStreamingAccessForServices resource-explorer-2:ListSupportedResourceTypes resource-explorer-2:ListViews resource-explorer-2:Search resource-explorer-2:UpdateIndexType resource-explorer-2:UpdateView

服务前缀	操作
resource-groups	resource-groups:CancelTagSyncTask
	resource-groups:GetAccountSettings
	resource-groups:GetGroup
	resource-groups:GetGroupConfiguration
	resource-groups:GetGroupQuery
	resource-groups:GetTagSyncTask
	resource-groups:GroupResources
	resource-groups:ListGroupResources
	resource-groups:ListGroupingStatuses
	resource-groups:ListGroups
	resource-groups:ListTagSyncTasks
	resource-groups:PutGroupConfiguration
	resource-groups:SearchResources
	resource-groups:StartTagSyncTask
	resource-groups:UngroupResources
	resource-groups:UpdateAccountSettings
	resource-groups:UpdateGroup
	resource-groups:UpdateGroupQuery

服务前缀	操作
robomaker	robomaker:BatchDeleteWorlds robomaker:BatchDescribeSimulationJob robomaker:CancelDeploymentJob robomaker:CancelSimulationJob robomaker:CancelSimulationJobBatch robomaker:CancelWorldExportJob robomaker:CancelWorldGenerationJob robomaker:CreateDeploymentJob robomaker:CreateFleet robomaker:CreateRobot robomaker:CreateRobotApplication robomaker:CreateRobotApplicationVersion robomaker:CreateSimulationApplication robomaker:CreateSimulationApplicationVersion robomaker:CreateSimulationJob robomaker:CreateWorldExportJob robomaker:CreateWorldGenerationJob robomaker:CreateWorldTemplate RoboMaker: DeleteFleet robomaker:DeleteRobot robomaker:DeleteRobotApplication

服务前缀	操作
	robomaker:DeleteSimulationApplication robomaker:DeleteWorldTemplate robomaker:DeregisterRobot robomaker:DescribeDeploymentJob robomaker:DescribeFleet robomaker:DescribeRobot robomaker:DescribeRobotApplication robomaker:DescribeSimulationApplication robomaker:DescribeSimulationJob robomaker:DescribeSimulationJobBatch robomaker:DescribeWorld robomaker:DescribeWorldExportJob robomaker:DescribeWorldGenerationJob robomaker:DescribeWorldTemplate robomaker:GetWorldTemplateBody robomaker:ListDeploymentJobs robomaker:ListFleets robomaker:ListRobotApplications robomaker:ListRobots robomaker:ListSimulationApplications robomaker:ListSimulationJobBatches

服务前缀	操作
	robomaker:ListSimulationJobs robomaker:ListWorldExportJobs robomaker:ListWorldGenerationJobs robomaker:ListWorldTemplates robomaker:ListWorlds robomaker:RegisterRobot robomaker:RestartSimulationJob robomaker:StartSimulationJobBatch robomaker:SyncDeploymentJob robomaker:UpdateRobotApplication robomaker:UpdateSimulationApplication robomaker:UpdateWorldTemplate

服务前缀	操作
rolesanywhere	rolesanywhere:CreateProfile
	rolesanywhere:CreateTrustAnchor
	rolesanywhere:DeleteAttributeMapping
	rolesanywhere:DeleteCrl
	rolesanywhere:DeleteProfile
	rolesanywhere:DeleteTrustAnchor
	rolesanywhere:DisableCrl
	rolesanywhere:DisableProfile
	rolesanywhere:DisableTrustAnchor
	rolesanywhere:EnableCrl
	rolesanywhere:EnableProfile
	rolesanywhere:EnableTrustAnchor
	rolesanywhere:GetCrl
	rolesanywhere:GetProfile
	rolesanywhere:GetSubject
	rolesanywhere:GetTrustAnchor
	rolesanywhere:ImportCrl
	rolesanywhere:ListCrls
	rolesanywhere:ListProfiles
	rolesanywhere:ListSubjects
	rolesanywhere:ListTrustAnchors

服务前缀	操作
	rolesanywhere:PutAttributeMapping
	rolesanywhere:PutNotificationSettings
	rolesanywhere:ResetNotificationSettings
	rolesanywhere:UpdateCrl
	rolesanywhere:UpdateProfile
	rolesanywhere:UpdateTrustAnchor

服务前缀	操作
route53	route53:ActivateKeySigningKey route53:AssociateVPCWithHostedZone route53:ChangeCidrCollection route53:ChangeResourceRecordSets route53:CreateCidrCollection route53:CreateHealthCheck route53:CreateHostedZone route53:CreateKeySigningKey route53:CreateQueryLoggingConfig route53:CreateReusableDelegationSet route53:CreateTrafficPolicy route53:CreateTrafficPolicyInstance route53:CreateTrafficPolicyVersion route53:CreateVPCAssociationAuthorization route53:DeactivateKeySigningKey route53:DeleteCidrCollection route53:DeleteHealthCheck route53:DeleteHostedZone route53:DeleteKeySigningKey route53:DeleteQueryLoggingConfig route53:DeleteReusableDelegationSet

服务前缀	操作
	route53:DeleteTrafficPolicy route53:DeleteTrafficPolicyInstance route53:DeleteVPCAssociationAuthorization route53:DisableHostedZoneDNSSEC route53:DisassociateVPCFromHostedZone route53:EnableHostedZoneDNSSEC route53:GetAccountLimit route53:GetChange route53:GetCheckerIpRanges route53:GetDNSSEC route53:GetGeoLocation route53:GetHealthCheck route53:GetHealthCheckCount route53:GetHealthCheckLastFailureReason route53:GetHealthCheckStatus route53:GetHostedZone route53:GetHostedZoneCount route53:GetHostedZoneLimit route53:GetQueryLoggingConfig route53:GetReusableDelegationSet route53:GetReusableDelegationSetLimit

服务前缀	操作
	route53:GetTrafficPolicy route53:GetTrafficPolicyInstance route53:GetTrafficPolicyInstanceCount route53:ListCidrBlocks route53:ListCidrCollections route53:ListCidrLocations route53:ListGeoLocations route53:ListHealthChecks route53:ListHostedZones route53:ListHostedZonesByName route53:ListHostedZonesByVPC route53:ListQueryLoggingConfigs route53:ListResourceRecordSets route53:ListReusableDelegationSets route53:ListTrafficPolicies route53:ListTrafficPolicyInstances route53:ListTrafficPolicyInstancesByHostedZone route53:ListTrafficPolicyInstancesByPolicy route53:ListTrafficPolicyVersions route53:ListVPCAssociationAuthorizations route53:TestDNSAnswer

服务前缀	操作
	route53:UpdateHealthCheck route53:UpdateHostedZoneComment route53:UpdateTrafficPolicyComment route53:UpdateTrafficPolicyInstance

服务前缀	操作
route53-recovery-control-config	route53-recovery-control-config:CreateCluster route53-recovery-control-config:CreateControlPanel route53-recovery-control-config:CreateRoutingControl route53-recovery-control-config:CreateSafetyRule route53-recovery-control-config:DeleteCluster route53-recovery-control-config:DeleteControlPanel route53-recovery-control-config:DeleteRoutingControl route53-recovery-control-config:DeleteSafetyRule route53-recovery-control-config:DescribeCluster route53-recovery-control-config:DescribeControlPanel route53-recovery-control-config:DescribeRoutingControl route53-recovery-control-config:DescribeSafetyRule route53-recovery-control-config:GetResourcePolicy route53-recovery-control-config:ListAssociatedRoute53HealthChecks route53-recovery-control-config:ListClusters route53-recovery-control-config:ListControlPanels route53-recovery-control-config:ListRoutingControls route53-recovery-control-config:ListSafetyRules route53-recovery-control-config:UpdateCluster route53-recovery-control-config:UpdateControlPanel

服务前缀	操作
	route53-recovery-control-config:UpdateRoutingControl route53-recovery-control-config:UpdateSafetyRule

服务前缀	操作
route53-recovery-readiness	route53-recovery-readiness:CreateCell route53-recovery-readiness:CreateCrossAccountAuthorization route53-recovery-readiness:CreateReadinessCheck route53-recovery-readiness:CreateRecoveryGroup route53-recovery-readiness:CreateResourceSet route53-recovery-readiness:DeleteCell route53-recovery-readiness:DeleteCrossAccountAuthorization route53-recovery-readiness:DeleteReadinessCheck route53-recovery-readiness:DeleteRecoveryGroup route53-recovery-readiness:DeleteResourceSet route53-recovery-readiness:GetArchitectureRecommendations route53-recovery-readiness:GetCell route53-recovery-readiness:GetCellReadinessSummary route53-recovery-readiness:GetReadinessCheck route53-recovery-readiness:GetReadinessCheckResourceStatus route53-recovery-readiness:GetReadinessCheckStatus route53-recovery-readiness:GetRecoveryGroup route53-recovery-readiness:GetRecoveryGroupReadinessSummary route53-recovery-readiness:GetResourceSet route53-recovery-readiness:ListCells route53-recovery-readiness:ListCrossAccountAuthorizations

服务前缀	操作
	route53-recovery-readiness:ListReadinessChecks route53-recovery-readiness:ListRecoveryGroups route53-recovery-readiness:ListResourceSets route53-recovery-readiness:ListRules route53-recovery-readiness:UpdateCell route53-recovery-readiness:UpdateReadinessCheck route53-recovery-readiness:UpdateRecoveryGroup route53-recovery-readiness:UpdateResourceSet

服务前缀	操作
route53resolver	route53resolver:AssociateFirewallRuleGroup
	route53resolver:AssociateResolverEndpointIpAddress
	route53resolver:AssociateResolverQueryLogConfig
	route53resolver:AssociateResolverRule
	route53resolver:CreateFirewallDomainList
	route53resolver:CreateFirewallRule
	route53resolver:CreateFirewallRuleGroup
	route53resolver:CreateResolverEndpoint
	route53resolver:CreateResolverQueryLogConfig
	route53resolver:CreateResolverRule
	route53resolver:DeleteFirewallDomainList
	route53resolver:DeleteFirewallRule
	route53resolver:DeleteFirewallRuleGroup
	route53resolver:DeleteOutpostResolver
	route53resolver:DeleteResolverEndpoint
	route53resolver:DeleteResolverQueryLogConfig
	route53resolver:DeleteResolverRule
	route53resolver:DisassociateFirewallRuleGroup
	route53resolver:DisassociateResolverEndpointIpAddress
	route53resolver:DisassociateResolverQueryLogConfig
	route53resolver:DisassociateResolverRule

服务前缀	操作
	route53resolver:GetFirewallConfig
	route53resolver:GetFirewallDomainList
	route53resolver:GetFirewallRuleGroup
	route53resolver:GetFirewallRuleGroupAssociation
	route53resolver:GetFirewallRuleGroupPolicy
	route53resolver:GetOutpostResolver
	route53resolver:GetResolverConfig
	route53resolver:GetResolverDnssecConfig
	route53resolver:GetResolverEndpoint
	route53resolver:GetResolverQueryLogConfig
	route53resolver:GetResolverQueryLogConfigAssociation
	route53resolver:GetResolverQueryLogConfigPolicy
	route53resolver:GetResolverRule
	route53resolver:GetResolverRuleAssociation
	route53resolver:GetResolverRulePolicy
	route53resolver:ImportFirewallDomains
	route53resolver:ListFirewallConfigs
	route53resolver:ListFirewallDomainLists
	route53resolver:ListFirewallDomains
	route53resolver:ListFirewallRuleGroupAssociations
	route53resolver:ListFirewallRuleGroups

服务前缀	操作
	route53resolver:ListFirewallRules
	route53resolver:ListOutpostResolvers
	route53resolver:ListResolverConfigs
	route53resolver:ListResolverDnssecConfigs
	route53resolver:ListResolverEndpointIpAddresses
	route53resolver:ListResolverEndpoints
	route53resolver:ListResolverQueryLogConfigAssociations
	route53resolver:ListResolverQueryLogConfigs
	route53resolver:ListResolverRuleAssociations
	route53resolver:ListResolverRules
	route53resolver:PutFirewallRuleGroupPolicy
	route53resolver:PutResolverQueryLogConfigPolicy
	route53resolver:UpdateFirewallConfig
	route53resolver:UpdateFirewallDomains
	route53resolver:UpdateFirewallRule
	route53resolver:UpdateFirewallRuleGroupAssociation
	route53resolver:UpdateOutpostResolver
	route53resolver:UpdateResolverConfig
	route53resolver:UpdateResolverDnssecConfig
	route53resolver:UpdateResolverEndpoint
	route53resolver:UpdateResolverRule

服务前缀	操作
rum	rum:BatchCreateRumMetricDefinitions rum:BatchDeleteRumMetricDefinitions rum:BatchGetRumMetricDefinitions rum:CreateAppMonitor rum:DeleteAppMonitor rum:DeleteResourcePolicy rum:DeleteRumMetricsDestination rum:GetAppMonitor rum:GetAppMonitorData rum:GetResourcePolicy rum:ListAppMonitors rum:ListRumMetricsDestinations rum:PutResourcePolicy rum:PutRumMetricsDestination rum:UpdateAppMonitor rum:UpdateRumMetricDefinition

服务前缀	操作
S3	s3:AssociateAccessGrantsIdentityCenter
	s3:CreateAccessGrant
	s3:CreateAccessGrantsInstance
	s3:CreateAccessGrantsLocation
	s3:CreateAccessPoint
	s3:CreateAccessPointForObjectLambda
	s3:CreateBucket
	s3:CreateBucketMetadataTableConfiguration
	s3:CreateJob
	s3:CreateMultiRegionAccessPoint
	s3:DeleteAccessGrant
	s3:DeleteAccessGrantsInstance
	s3:DeleteAccessGrantsInstanceResourcePolicy
	s3:DeleteAccessGrantsLocation
	s3:DeleteAccessPoint
	s3:DeleteAccessPointForObjectLambda
	s3:DeleteAccessPointPolicy
	s3:DeleteAccessPointPolicyForObjectLambda
	s3:DeleteBucket
	s3:DeleteBucketMetadataTableConfiguration
	s3:DeleteBucketPolicy

服务前缀	操作
	s3:DeleteBucketWebsite
	s3:DeleteMultiRegionAccessPoint
	s3:DeleteStorageLensConfiguration
	s3:DescribeJob
	s3:DescribeMultiRegionAccessPointOperation
	s3:DissociateAccessGrantsIdentityCenter
	s3:GetAccelerateConfiguration
	s3:GetAccessGrant
	s3:GetAccessGrantsInstance
	s3:GetAccessGrantsInstanceForPrefix
	s3:GetAccessGrantsInstanceResourcePolicy
	s3:GetAccessGrantsLocation
	s3:GetAccessPoint
	s3:GetAccessPointConfigurationForObjectLambda
	s3:GetAccessPointForObjectLambda
	s3:GetAccessPointPolicy
	s3:GetAccessPointPolicyForObjectLambda
	s3:GetAccessPointPolicyStatus
	s3:GetAccessPointPolicyStatusForObjectLambda
	s3:GetAccountPublicAccessBlock
	s3:GetAnalyticsConfiguration

服务前缀	操作
	s3:GetBucketAbac
	s3:GetBucketAcl
	s3:GetBucketCORS
	s3:GetBucketLocation
	s3:GetBucketLogging
	s3:GetBucketNotification
	s3:GetBucketObjectLockConfiguration
	s3:GetBucketOwnershipControls
	s3:GetBucketPolicy
	s3:GetBucketPolicyStatus
	s3:GetBucketPublicAccessBlock
	s3:GetBucketRequestPayment
	s3:GetBucketVersioning
	s3:GetBucketWebsite
	s3:GetDataAccess
	s3:GetEncryptionConfiguration
	s3:GetIntelligentTieringConfiguration
	s3:GetInventoryConfiguration
	s3:GetLifecycleConfiguration
	s3:GetMetricsConfiguration
	s3:GetMultiRegionAccessPoint

服务前缀	操作
	s3:GetMultiRegionAccessPointPolicy
	s3:GetMultiRegionAccessPointPolicyStatus
	s3:GetMultiRegionAccessPointRoutes
	s3:GetReplicationConfiguration
	s3:GetStorageLensConfiguration
	s3:GetStorageLensDashboard
	s3:ListAccessGrants
	s3:ListAccessGrantsInstances
	s3:ListAccessGrantsLocations
	s3:ListAccessPoints
	s3:ListAccessPointsForObjectLambda
	s3:ListAllMyBuckets
	s3:ListBucketMultipartUploads
	s3:ListCallerAccessGrants
	s3:ListJobs
	s3:ListMultiRegionAccessPoints
	s3:ListStorageLensConfigurations
	s3:PutAccelerateConfiguration
	s3:PutAccessGrantsInstanceResourcePolicy
	s3:PutAccessPointConfigurationForObjectLambda
	s3:PutAccessPointPolicy

服务前缀	操作
	s3:PutAccessPointPolicyForObjectLambda
	s3:PutAccountPublicAccessBlock
	s3:PutAnalyticsConfiguration
	s3:PutBucketAbac
	s3:PutBucketAcl
	s3:PutBucketCORS
	s3:PutBucketLogging
	s3:PutBucketNotification
	s3:PutBucketObjectLockConfiguration
	s3:PutBucketOwnershipControls
	s3:PutBucketPolicy
	s3:PutBucketPublicAccessBlock
	s3:PutBucketRequestPayment
	s3:PutBucketVersioning
	s3:PutBucketWebsite
	s3:PutEncryptionConfiguration
	s3:PutIntelligentTieringConfiguration
	s3:PutInventoryConfiguration
	s3:PutLifecycleConfiguration
	s3:PutMetricsConfiguration
	s3:PutMultiRegionAccessPointPolicy

服务前缀	操作
	s3:PutReplicationConfiguration s3:PutStorageLensConfiguration s3:SubmitMultiRegionAccessPointRoutes s3:UpdateAccessGrantsLocation s3:UpdateBucketMetadataJournalTableConfiguration s3:UpdateJobPriority s3:UpdateJobStatus
s3-outposts	s3-outposts:CreateEndpoint s3-outposts:DeleteEndpoint s3-outposts:ListEndpoints s3-outposts:ListOutpostsWithS3 s3-outposts:ListSharedEndpoints

服务前缀	操作
sagemaker-geospatial	sagemaker-geospatial:DeleteEarthObservationJob sagemaker-geospatial:DeleteVectorEnrichmentJob sagemaker-geospatial:ExportEarthObservationJob sagemaker-geospatial:ExportVectorEnrichmentJob sagemaker-geospatial:GetEarthObservationJob sagemaker-geospatial:GetRasterDataCollection sagemaker-geospatial:GetTile sagemaker-geospatial:GetVectorEnrichmentJob sagemaker-geospatial:ListEarthObservationJobs sagemaker-geospatial:ListRasterDataCollections sagemaker-geospatial:ListVectorEnrichmentJobs sagemaker-geospatial:SearchRasterDataCollection sagemaker-geospatial:StartEarthObservationJob sagemaker-geospatial:StartVectorEnrichmentJob sagemaker-geospatial:StopEarthObservationJob sagemaker-geospatial:StopVectorEnrichmentJob

服务前缀	操作
savingsplans	savingsplans:CreateSavingsPlan savingsplans:DeleteQueuedSavingsPlan savingsplans:DescribeSavingsPlanRates savingsplans:DescribeSavingsPlans savingsplans:DescribeSavingsPlansOfferingRates savingsplans:DescribeSavingsPlansOfferings savingsplans:ReturnSavingsPlan

服务前缀	操作
schemas	schemas:CreateDiscoverer
	schemas:CreateRegistry
	schemas:CreateSchema
	schemas:DeleteDiscoverer
	schemas:DeleteRegistry
	schemas:DeleteResourcePolicy
	schemas:DeleteSchema
	schemas:DeleteSchemaVersion
	schemas:DescribeCodeBinding
	schemas:DescribeDiscoverer
	schemas:DescribeRegistry
	schemas:DescribeSchema
	schemas:ExportSchema
	schemas:GetCodeBindingSource
	schemas:GetDiscoveredSchema
	schemas:GetResourcePolicy
	schemas:ListDiscoverers
	schemas:ListRegistries
	schemas:ListSchemaVersions
	schemas:ListSchemas
	schemas:PutCodeBinding

服务前缀	操作
	schemas:PutResourcePolicy schemas:SearchSchemas schemas:StartDiscoverer schemas:StopDiscoverer schemas:UpdateDiscoverer schemas:UpdateRegistry schemas:UpdateSchema
sdb	sdb:CreateDomain sdb:DeleteDomain sdb:DomainMetadata sdb:ListDomains

服务前缀	操作
secretsmanager	secretsmanager:CancelRotateSecret secretsmanager:CreateSecret secretsmanager:DeleteResourcePolicy secretsmanager:DeleteSecret secretsmanager:DescribeSecret secretsmanager:GetRandomPassword secretsmanager:GetResourcePolicy secretsmanager:GetSecretValue secretsmanager:ListSecretVersionIds secretsmanager:ListSecrets secretsmanager:PutResourcePolicy secretsmanager:PutSecretValue secretsmanager:RemoveRegionsFromReplication secretsmanager:ReplicateSecretToRegions secretsmanager:RestoreSecret secretsmanager:RotateSecret secretsmanager:StopReplicationToReplica secretsmanager:UpdateSecret secretsmanager:ValidateResourcePolicy

服务前缀	操作
securityhub	securityhub:AcceptAdministratorInvitation securityhub:AcceptInvitation securityhub:BatchDeleteAutomationRules securityhub:BatchDisableStandards securityhub:BatchEnableStandards securityhub:BatchGetAutomationRules securityhub:BatchGetConfigurationPolicyAssociations securityhub:BatchGetSecurityControls securityhub:BatchGetStandardsControlAssociations securityhub:BatchImportFindings securityhub:BatchUpdateAutomationRules securityhub:BatchUpdateFindings securityhub:BatchUpdateStandardsControlAssociations securityhub:ConnectorRegistrationsV2 securityhub:CreateActionTarget securityhub:CreateAggregatorV2 securityhub:CreateAutomationRule securityhub:CreateAutomationRuleV2 securityhub:CreateConfigurationPolicy securityhub:CreateConnectorV2 securityhub:CreateFindingAggregator

服务前缀	操作
	securityhub:CreateInsight securityhub:CreateMembers securityhub:CreateTicketV2 securityhub:DeclineInvitations securityhub:DeleteActionTarget securityhub:DeleteAggregatorV2 securityhub:DeleteAutomationRuleV2 securityhub:DeleteConfigurationPolicy securityhub:DeleteConnectorV2 securityhub:DeleteFindingAggregator securityhub:DeleteInsight securityhub:DeleteInvitations securityhub:DeleteMembers securityhub:DescribeActionTargets securityhub:DescribeHub securityhub:DescribeOrganizationConfiguration securityhub:DescribeProducts securityhub:DescribeSecurityHubV2 securityhub:DescribeStandards securityhub:DisableImportFindingsForProduct securityhub:DisableOrganizationAdminAccount

服务前缀	操作
	securityhub:DisableSecurityHub securityhub:DisableSecurityHubV2 securityhub:DisassociateFromAdministratorAccount securityhub:DisassociateFromMasterAccount securityhub:DisassociateMembers securityhub:EnableImportFindingsForProduct securityhub:EnableOrganizationAdminAccount securityhub:EnableSecurityHub securityhub:GetAdministratorAccount securityhub:GetAggregatorV2 securityhub:GetAutomationRuleV2 securityhub:GetConfigurationPolicy securityhub:GetConfigurationPolicyAssociation securityhub:GetConnectorV2 securityhub:GetEnabledStandards securityhub:GetFindingAggregator securityhub:GetFindingHistory securityhub:GetFindings securityhub:GetInsightResults securityhub:GetInsights securityhub:GetInvitationsCount

服务前缀	操作
	securityhub:GetMasterAccount securityhub:GetMembers securityhub:GetSecurityControlDefinition securityhub:InviteMembers securityhub:ListAggregatorsV2 securityhub:ListAutomationRules securityhub:ListAutomationRulesV2 securityhub:ListConfigurationPolicies securityhub:ListConfigurationPolicyAssociations securityhub:ListConnectorsV2 securityhub:ListEnabledProductsForImport securityhub:ListFindingAggregators securityhub:ListInvitations securityhub:ListMembers securityhub:ListOrganizationAdminAccounts securityhub:ListSecurityControlDefinitions securityhub:ListStandardsControlAssociations securityhub:StartConfigurationPolicyAssociation securityhub:StartConfigurationPolicyDisassociation securityhub:UpdateActionTarget securityhub:UpdateAggregatorV2

服务前缀	操作
	securityhub:UpdateAutomationRuleV2 securityhub:UpdateConfigurationPolicy securityhub:UpdateConnectorV2 securityhub:UpdateFindingAggregator securityhub:UpdateFindings securityhub:UpdateInsight securityhub:UpdateOrganizationConfiguration securityhub:UpdateSecurityControl securityhub:UpdateSecurityHubConfiguration

服务前缀	操作
securitylake	securitylake:CreateAwsLogSource securitylake:CreateCustomLogSource securitylake:CreateDataLakeExceptionSubscription securitylake:CreateDataLakeOrganizationConfiguration securitylake:CreateSubscriber securitylake:CreateSubscriberNotification securitylake:DeleteAwsLogSource securitylake:DeleteCustomLogSource securitylake:DeleteDataLakeExceptionSubscription securitylake:DeleteDataLakeOrganizationConfiguration securitylake:DeleteSubscriber securitylake:DeleteSubscriberNotification securitylake:DeregisterDataLakeDelegatedAdministrator securitylake:GetDataLakeExceptionSubscription securitylake:GetDataLakeOrganizationConfiguration securitylake:GetDataLakeSources securitylake:GetSubscriber securitylake:ListDataLakes securitylake:ListLogSources securitylake:ListSubscribers securitylake:RegisterDataLakeDelegatedAdministrator

服务前缀	操作
	securitylake:UpdateDataLakeExceptionSubscription securitylake:UpdateSubscriber securitylake:UpdateSubscriberNotification
serverlessrepo	serverlessrepo:CreateApplication serverlessrepo:CreateApplicationVersion serverlessrepo:CreateCloudFormationChangeSet serverlessrepo:CreateCloudFormationTemplate serverlessrepo:DeleteApplication serverlessrepo:GetApplication serverlessrepo:GetApplicationPolicy serverlessrepo:GetCloudFormationTemplate serverlessrepo:ListApplicationDependencies serverlessrepo:ListApplicationVersions serverlessrepo:ListApplications serverlessrepo:PutApplicationPolicy serverlessrepo:UnshareApplication serverlessrepo:UpdateApplication

服务前缀	操作
servicecatalog	servicecatalog:AcceptPortfolioShare servicecatalog:AssociateBudgetWithResource servicecatalog:AssociatePrincipalWithPortfolio servicecatalog:AssociateProductWithPortfolio servicecatalog:AssociateServiceActionWithProvisioningArtifact servicecatalog:BatchAssociateServiceActionWithProvisioningArtifact servicecatalog:BatchDisassociateServiceActionFromProvisioningArtifact servicecatalog:CopyProduct servicecatalog:CreateAttributeGroup servicecatalog:CreateConstraint servicecatalog:CreatePortfolio servicecatalog:CreatePortfolioShare servicecatalog:CreateProduct servicecatalog:CreateProvisionedProductPlan servicecatalog:CreateProvisioningArtifact servicecatalog:CreateServiceAction servicecatalog:DeleteAttributeGroup servicecatalog:DeleteConstraint servicecatalog:DeletePortfolio servicecatalog:DeletePortfolioShare

服务前缀	操作
	servicecatalog:DeleteProduct
	servicecatalog:DeleteProvisionedProductPlan
	servicecatalog:DeleteProvisioningArtifact
	servicecatalog:DeleteServiceAction
	servicecatalog:DescribeConstraint
	servicecatalog:DescribeCopyProductStatus
	servicecatalog:DescribePortfolio
	servicecatalog:DescribePortfolioShareStatus
	servicecatalog:DescribePortfolioShares
	servicecatalog:DescribeProduct
	servicecatalog:DescribeProductAsAdmin
	servicecatalog:DescribeProductView
	servicecatalog:DescribeProvisionedProduct
	servicecatalog:DescribeProvisionedProductPlan
	servicecatalog:DescribeProvisioningArtifact
	servicecatalog:DescribeProvisioningParameters
	servicecatalog:DescribeRecord
	servicecatalog:DescribeServiceAction
	servicecatalog:DescribeServiceActionExecutionParameters
	servicecatalog:DisableAWSOrganizationsAccess
	servicecatalog:DisassociateBudgetFromResource

服务前缀	操作
	servicecatalog:DisassociatePrincipalFromPortfolio
	servicecatalog:DisassociateProductFromPortfolio
	servicecatalog:DisassociateServiceActionFromProvisioningArtifact
	servicecatalog:EnableAWSOrganizationsAccess
	servicecatalog:ExecuteProvisionedProductPlan
	servicecatalog:ExecuteProvisionedProductServiceAction
	servicecatalog:GetAWSOrganizationsAccessStatus
	servicecatalog:GetProvisionedProductOutputs
	servicecatalog:ImportAsProvisionedProduct
	servicecatalog:ListAcceptedPortfolioShares
	servicecatalog:ListAttributeGroups
	servicecatalog:ListBudgetsForResource
	servicecatalog:ListConstraintsForPortfolio
	servicecatalog:ListLaunchPaths
	servicecatalog:ListOrganizationPortfolioAccess
	servicecatalog:ListPortfolioAccess
	servicecatalog:ListPortfolios
	servicecatalog:ListPortfoliosForProduct
	servicecatalog:ListPrincipalsForPortfolio
	servicecatalog:ListProvisionedProductPlans
	servicecatalog:ListProvisioningArtifacts

服务前缀	操作
	servicecatalog:ListProvisioningArtifactsForServiceAction
	servicecatalog:ListRecordHistory
	servicecatalog:ListServiceActions
	servicecatalog:ListServiceActionsForProvisioningArtifact
	servicecatalog:ListStackInstancesForProvisionedProduct
	servicecatalog:NotifyProvisionProductEngineWorkflowResult
	servicecatalog:NotifyTerminateProvisionedProductEngine WorkflowResult
	servicecatalog:NotifyUpdateProvisionedProductEngineWor kflowResult
	servicecatalog:ProvisionProduct
	servicecatalog:RejectPortfolioShare
	servicecatalog:ScanProvisionedProducts
	servicecatalog:SearchProducts
	servicecatalog:SearchProductsAsAdmin
	servicecatalog:SearchProvisionedProducts
	servicecatalog:TerminateProvisionedProduct
	servicecatalog:UpdateConstraint
	servicecatalog:UpdatePortfolio
	servicecatalog:UpdatePortfolioShare
	servicecatalog:UpdateProduct
	servicecatalog:UpdateProvisionedProduct

服务前缀	操作
	servicecatalog:UpdateProvisionedProductProperties servicecatalog:UpdateProvisioningArtifact servicecatalog:UpdateServiceAction

服务前缀	操作
servicediscovery	servicediscovery:CreateHttpNamespace servicediscovery:CreatePrivateDnsNamespace servicediscovery:CreatePublicDnsNamespace servicediscovery:CreateService servicediscovery:DeleteNamespace servicediscovery:DeleteService servicediscovery:DeleteServiceAttributes servicediscovery:DeregisterInstance servicediscovery:GetInstance servicediscovery:GetInstancesHealthStatus servicediscovery:GetNamespace servicediscovery:GetOperation servicediscovery:GetService servicediscovery:ListInstances servicediscovery:ListNamespaces servicediscovery:ListOperations servicediscovery:ListServices servicediscovery:RegisterInstance servicediscovery:UpdateHttpNamespace servicediscovery:UpdateInstanceCustomHealthStatus servicediscovery:UpdatePrivateDnsNamespace

服务前缀	操作
	servicediscovery:UpdatePublicDnsNamespace servicediscovery:UpdateService servicediscovery:UpdateServiceAttributes

服务前缀	操作
servicequotas	servicequotas:AssociateServiceQuotaTemplate servicequotas:CreateSupportCase servicequotas:DeleteServiceQuotaIncreaseRequestFromTemplate servicequotas:DisassociateServiceQuotaTemplate servicequotas:GetAWSDefaultServiceQuota servicequotas:GetAssociationForServiceQuotaTemplate servicequotas:GetAutoManagementConfiguration servicequotas:GetQuotaUtilizationReport servicequotas:GetRequestedServiceQuotaChange servicequotas:GetServiceQuota servicequotas:GetServiceQuotaIncreaseRequestFromTemplate servicequotas:ListAWSDefaultServiceQuotas servicequotas:ListRequestedServiceQuotaChangeHistory servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota servicequotas:ListServiceQuotaIncreaseRequestsInTemplate servicequotas:ListServiceQuotas servicequotas:ListServices servicequotas:PutServiceQuotaIncreaseRequestIntoTemplate servicequotas:RequestServiceQuotaIncrease servicequotas:StartAutoManagement servicequotas:StartQuotaUtilizationReport

服务前缀	操作
	servicequotas:StopAutoManagement servicequotas:UpdateAutoManagement

服务前缀	操作
ses	ses:BatchGetMetricData ses:CloneReceiptRuleSet ses:CreateAddonInstance ses:CreateAddonSubscription ses:CreateAddressList ses:CreateAddressListImportJob ses:CreateArchive ses:CreateConfigurationSet ses:CreateConfigurationSetEventDestination ses:CreateConfigurationSetTrackingOptions ses:CreateContact ses:CreateContactList ses:CreateCustomVerificationEmailTemplate ses:CreateDedicatedIpPool ses:CreateDeliverabilityTestReport ses:CreateEmailIdentity ses:CreateEmailIdentityPolicy ses:CreateEmailTemplate ses:CreateImportJob ses:CreateIngressPoint ses:CreateMultiRegionEndpoint

服务前缀	操作
	ses:CreateReceiptFilter ses:CreateReceiptRule ses:CreateReceiptRuleSet ses:CreateRelay ses:CreateRuleSet ses:CreateTemplate ses:CreateTenant ses:CreateTenantResourceAssociation ses:CreateTrafficPolicy ses:DeleteAddonInstance ses:DeleteAddonSubscription ses:DeleteAddressList ses:DeleteArchive ses:DeleteConfigurationSet ses:DeleteConfigurationSetEventDestination ses:DeleteConfigurationSetTrackingOptions ses:DeleteContact ses:DeleteContactList ses:DeleteCustomVerificationEmailTemplate ses:DeleteDedicatedIpPool ses:DeleteEmailIdentity

服务前缀	操作
	ses:DeleteEmailIdentityPolicy ses:DeleteEmailTemplate ses:DeleteIdentity ses:DeleteIdentityPolicy ses:DeleteIngressPoint ses:DeleteMultiRegionEndpoint ses:DeleteReceiptFilter ses:DeleteReceiptRule ses:DeleteReceiptRuleSet ses:DeleteRelay ses:DeleteRuleSet ses:DeleteSuppressedDestination ses:DeleteTemplate ses:DeleteTenant ses:DeleteTenantResourceAssociation ses:DeleteTrafficPolicy ses:DeleteVerifiedEmailAddress ses:DeregisterMemberFromAddressList ses:DescribeActiveReceiptRuleSet ses:DescribeConfigurationSet ses:DescribeReceiptRule

服务前缀	操作
	ses:DescribeReceiptRuleSet ses:GetAccount ses:GetAccountSendingEnabled ses:GetAddonInstance ses:GetAddonSubscription ses:GetAddressList ses:GetArchive ses:GetArchiveExport ses:GetArchiveMessage ses:GetArchiveMessageContent ses:GetArchiveSearch ses:GetArchiveSearchResults ses:GetBlacklistReports ses:GetConfigurationSet ses:GetConfigurationSetEventDestinations ses:GetContact ses:GetContactList ses:GetCustomVerificationEmailTemplate ses:GetDedicatedIp ses:GetDedicatedIpPool ses:GetDedicatedIps

服务前缀	操作
	ses:GetDeliverabilityDashboardOptions ses:GetDeliverabilityTestReport ses:GetDomainDeliverabilityCampaign ses:GetDomainStatisticsReport ses:GetEmailAddressInsights ses:GetEmailIdentity ses:GetEmailIdentityPolicies ses:GetEmailTemplate ses:GetIdentityDkimAttributes ses:GetIdentityMailFromDomainAttributes ses:GetIdentityNotificationAttributes ses:GetIdentityPolicies ses:GetIdentityVerificationAttributes ses:GetImportJob ses:GetIngressPoint ses:GetMemberOfAddressList ses:GetMessageInsights ses:GetMultiRegionEndpoint ses:GetRelay ses:GetRuleSet ses:GetSendQuota

服务前缀	操作
	ses:GetSendStatistics ses:GetSuppressedDestination ses:GetTemplate ses:GetTenant ses:GetTrafficPolicy ses:ListAddonInstances ses:ListAddonSubscriptions ses:ListAddressListImportJobs ses:ListAddressLists ses:ListArchiveExports ses:ListArchiveSearches ses:ListArchives ses:ListConfigurationSets ses:ListContactLists ses:ListContacts ses:ListCustomVerificationEmailTemplates ses:ListDedicatedIpPools ses:ListDeliverabilityTestReports ses:ListDomainDeliverabilityCampaigns ses:ListEmailIdentities ses:ListEmailTemplates

服务前缀	操作
	ses:ListExportJobs ses:ListIdentities ses:ListIdentityPolicies ses:ListImportJobs ses:ListIngressPoints ses:ListMembersOfAddressList ses:ListMultiRegionEndpoints ses:ListReceiptFilters ses:ListReceiptRuleSets ses:ListRecommendations ses:ListRelays ses:ListReputationEntities ses:ListResourceTenants ses:ListRuleSets ses:ListSuppressedDestinations ses:ListTemplates ses:ListTenantResources ses:ListTenants ses:ListTrafficPolicies ses:ListVerifiedEmailAddresses ses:PutAccountDedicatedIpWarmupAttributes

服务前缀	操作
	ses:PutAccountDetails ses:PutAccountSendingAttributes ses:PutAccountSuppressionAttributes ses:PutAccountVdmAttributes ses:PutConfigurationSetArchivingOptions ses:PutConfigurationSetDeliveryOptions ses:PutConfigurationSetReputationOptions ses:PutConfigurationSetSendingOptions ses:PutConfigurationSetSuppressionOptions ses:PutConfigurationSetTrackingOptions ses:PutConfigurationSetVdmOptions ses:PutDedicatedIpInPool ses:PutDedicatedIpPoolScalingAttributes ses:PutDedicatedIpWarmupAttributes ses:PutDeliverabilityDashboardOption ses:PutEmailIdentityConfigurationSetAttributes ses:PutEmailIdentityDkimAttributes ses:PutEmailIdentityDkimSigningAttributes ses:PutEmailIdentityFeedbackAttributes ses:PutEmailIdentityMailFromAttributes ses:PutIdentityPolicy

服务前缀	操作
	ses:PutSuppressedDestination ses:RegisterMemberToAddressList ses:ReorderReceiptRuleSet ses:SendBounce ses:SendCustomVerificationEmail ses:SetActiveReceiptRuleSet ses:SetIdentityDkimEnabled ses:SetIdentityFeedbackForwardingEnabled ses:SetIdentityHeadersInNotificationsEnabled ses:SetIdentityMailFromDomain ses:SetIdentityNotificationTopic ses:SetReceiptRulePosition ses:StartArchiveExport ses:StartArchiveSearch ses:StopArchiveExport ses:StopArchiveSearch ses:TestRenderEmailTemplate ses:TestRenderTemplate ses:UpdateAccountSendingEnabled ses:UpdateArchive ses:UpdateConfigurationSetEventDestination

服务前缀	操作
	ses:UpdateConfigurationSetReputationMetricsEnabled ses:UpdateConfigurationSetSendingEnabled ses:UpdateConfigurationSetTrackingOptions ses:UpdateContact ses:UpdateContactList ses:UpdateCustomVerificationEmailTemplate ses:UpdateEmailIdentityPolicy ses:UpdateEmailTemplate ses:UpdateIngressPoint ses:UpdateReceiptRule ses:UpdateRelay ses:UpdateRuleSet ses:UpdateTemplate ses:UpdateTrafficPolicy ses:VerifyDomainDkim ses:VerifyDomainIdentity ses:VerifyEmailAddress ses:VerifyEmailIdentity

服务前缀	操作
shield	shield:AssociateDRTLogBucket
	shield:AssociateHealthCheck
	shield:AssociateProactiveEngagementDetails
	shield:CreateProtection
	shield:CreateProtectionGroup
	shield:CreateSubscription
	shield:DeleteProtection
	shield:DeleteProtectionGroup
	shield:DeleteSubscription
	shield:DescribeAttack
	shield:DescribeAttackStatistics
	shield:DescribeDRTAccess
	shield:DescribeEmergencyContactSettings
	shield:DescribeProtection
	shield:DescribeProtectionGroup
	shield:DescribeSubscription
	shield:DisableApplicationLayerAutomaticResponse
	shield:DisableProactiveEngagement
	shield:DisassociateDRTLogBucket
	shield:DisassociateDRTRole
	shield:DisassociateHealthCheck

服务前缀	操作
	shield:EnableApplicationLayerAutomaticResponse
	shield:EnableProactiveEngagement
	shield:GetSubscriptionState
	shield:ListAttacks
	shield:ListProtectionGroups
	shield:ListProtections
	shield:ListResourcesInProtectionGroup
	shield:UpdateApplicationLayerAutomaticResponse
	shield:UpdateEmergencyContactSettings
	shield:UpdateProtectionGroup
	shield:UpdateSubscription

服务前缀	操作
signer	signer:AddProfilePermission signer:CancelSigningProfile signer:DescribeSigningJob signer:GetSigningPlatform signer:GetSigningProfile signer:ListProfilePermissions signer:ListSigningJobs signer:ListSigningPlatforms signer:ListSigningProfiles signer:PutSigningProfile signer:RemoveProfilePermission signer:RevokeSignature signer:RevokeSigningProfile signer:SignPayload signer:StartSigningJob

服务前缀	操作
simspaceweaver	simspaceweaver:CreateSnapshot simspaceweaver:DeleteApp simspaceweaver:DeleteSimulation simspaceweaver:DescribeApp simspaceweaver:DescribeSimulation simspaceweaver:ListApps simspaceweaver:ListSimulations simspaceweaver:StartApp simspaceweaver:StartClock simspaceweaver:StartSimulation simspaceweaver:StopApp simspaceweaver:StopClock simspaceweaver:StopSimulation

服务前缀	操作
sms	sms:CreateApp sms:CreateReplicationJob sms:DeleteApp sms:DeleteAppLaunchConfiguration sms:DeleteAppReplicationConfiguration sms:DeleteAppValidationConfiguration sms:DeleteReplicationJob sms:DeleteServerCatalog sms:DisassociateConnector sms:GenerateChangeSet sms:GenerateTemplate sms:GetApp sms:GetAppLaunchConfiguration sms:GetAppReplicationConfiguration sms:GetAppValidationConfiguration sms:GetAppValidationOutput sms:GetConnectors sms:GetReplicationJobs sms:GetReplicationRuns sms:GetServers sms:ImportAppCatalog

服务前缀	操作
	sms:ImportServerCatalog sms:LaunchApp sms:ListApps sms:NotifyAppValidationOutput sms:PutAppLaunchConfiguration sms:PutAppReplicationConfiguration sms:PutAppValidationConfiguration sms:StartAppReplication sms:StartOnDemandAppReplication sms:StartOnDemandReplicationRun sms:StopAppReplication sms:TerminateApp sms:UpdateApp sms:UpdateReplicationJob

服务前缀	操作
sms-voice	sms-voice:AssociateProtectConfiguration sms-voice:CreateConfigurationSet sms-voice:CreateConfigurationSetEventDestination sms-voice:CreateEventDestination sms-voice:CreateOptOutList sms-voice:CreatePool sms-voice:CreateProtectConfiguration sms-voice:CreateRegistration sms-voice:CreateRegistrationAssociation sms-voice:CreateRegistrationAttachment sms-voice:CreateRegistrationVersion sms-voice:CreateVerifiedDestinationNumber sms-voice:DeleteAccountDefaultProtectConfiguration sms-voice:DeleteConfigurationSet sms-voice:DeleteConfigurationSetEventDestination sms-voice:DeleteDefaultMessageType sms-voice:DeleteDefaultSenderId sms-voice:DeleteEventDestination sms-voice:DeleteKeyword sms-voice:DeleteMediaMessageSpendLimitOverride sms-voice:DeleteOptOutList

服务前缀	操作
	sms-voice:DeleteOptedOutNumber
	sms-voice:DeletePool
	sms-voice:DeleteProtectConfiguration
	sms-voice:DeleteProtectConfigurationRuleSetNumberOverride
	sms-voice:DeleteRegistration
	sms-voice:DeleteRegistrationAttachment
	sms-voice:DeleteResourcePolicy
	sms-voice:DeleteTextMessageSpendLimitOverride
	sms-voice:DeleteVerifiedDestinationNumber
	sms-voice:DeleteVoiceMessageSpendLimitOverride
	sms-voice:DescribeAccountAttributes
	sms-voice:DescribeAccountLimits
	sms-voice:DescribeConfigurationSets
	sms-voice:DescribeKeywords
	sms-voice:DescribeOptOutLists
	sms-voice:DescribeOptedOutNumbers
	sms-voice:DescribePhoneNumbers
	sms-voice:DescribePools
	sms-voice:DescribeProtectConfigurations
	sms-voice:DescribeRegistrationAttachments
	sms-voice:DescribeRegistrationFieldDefinitions

服务前缀	操作
	sms-voice:DescribeRegistrationFieldValues
	sms-voice:DescribeRegistrationSectionDefinitions
	sms-voice:DescribeRegistrationTypeDefinitions
	sms-voice:DescribeRegistrationVersions
	sms-voice:DescribeRegistrations
	sms-voice:DescribeSenderIds
	sms-voice:DescribeSpendLimits
	sms-voice:DescribeVerifiedDestinationNumbers
	sms-voice:DisassociateOriginationIdentity
	sms-voice:DisassociateProtectConfiguration
	sms-voice:DiscardRegistrationVersion
	sms-voice:GetConfigurationSetEventDestinations
	sms-voice:GetProtectConfigurationCountryRuleSet
	sms-voice:GetResourcePolicy
	sms-voice:ListConfigurationSets
	sms-voice:ListPoolOriginationIdentities
	sms-voice:ListProtectConfigurationRuleSetNumberOverrides
	sms-voice:ListRegistrationAssociations
	sms-voice:PutKeyword
	sms-voice:PutOptedOutNumber
	sms-voice:PutProtectConfigurationRuleSetNumberOverride

服务前缀	操作
	sms-voice:PutResourcePolicy
	sms-voice:ReleasePhoneNumber
	sms-voice:ReleaseSenderId
	sms-voice:RequestPhoneNumber
	sms-voice:RequestSenderId
	sms-voice:SendDestinationNumberVerificationCode
	sms-voice:SetAccountDefaultProtectConfiguration
	sms-voice:SetDefaultMessageFeedbackEnabled
	sms-voice:SetDefaultMessageType
	sms-voice:SetDefaultSenderId
	sms-voice:SetMediaMessageSpendLimitOverride
	sms-voice:SetTextMessageSpendLimitOverride
	sms-voice:SetVoiceMessageSpendLimitOverride
	sms-voice:SubmitRegistrationVersion
	sms-voice:UpdateConfigurationSetEventDestination
	sms-voice:UpdateEventDestination
	sms-voice:UpdatePhoneNumber
	sms-voice:UpdatePool
	sms-voice:UpdateProtectConfiguration
	sms-voice:UpdateProtectConfigurationCountryRuleSet
	sms-voice:UpdateSenderId

服务前缀	操作
snowball	snowball:CancelCluster snowball:CancelJob snowball:CreateAddress snowball:CreateCluster snowball:CreateJob snowball:CreateLongTermPricing snowball:CreateReturnShippingLabel snowball:DescribeAddress snowball:DescribeAddresses snowball:DescribeCluster snowball:DescribeJob snowball:DescribeReturnShippingLabel snowball:GetJobManifest snowball:GetJobUnlockCode snowball:GetSnowballUsage snowball:GetSoftwareUpdates snowball:ListClusterJobs snowball:ListClusters snowball:ListCompatibleImages snowball:ListJobs snowball:ListLongTermPricing

服务前缀	操作
	snowball:ListPickupLocations snowball:ListServiceVersions snowball:UpdateCluster snowball:UpdateJob snowball:UpdateJobShipmentState snowball:UpdateLongTermPricing
sqs	sqs:AddPermission sqs:CancelMessageMoveTask sqs:CreateQueue sqs:DeleteQueue sqs:PurgeQueue sqs:RemovePermission sqs:SetQueueAttributes

服务前缀	操作
ssm	ssm:AssociateOpsItemRelatedItem ssm:CancelCommand ssm:CancelMaintenanceWindowExecution ssm:CreateActivation ssm:CreateAssociation ssm:CreateAssociationBatch ssm:CreateDocument ssm:CreateMaintenanceWindow ssm:CreateOpsItem ssm:CreateOpsMetadata ssm:CreatePatchBaseline ssm:CreateResourceDataSync ssm:DeleteActivation ssm:DeleteAssociation ssm:DeleteDocument ssm:DeleteInventory ssm:DeleteMaintenanceWindow ssm:DeleteOpsItem ssm:DeleteOpsMetadata ssm:DeleteParameter ssm:DeleteParameters

服务前缀	操作
	ssm:DeletePatchBaseline ssm:DeleteResourceDataSync ssm:DeleteResourcePolicy ssm:DeregisterManagedInstance ssm:DeregisterPatchBaselineForPatchGroup ssm:DeregisterTargetFromMaintenanceWindow ssm:DeregisterTaskFromMaintenanceWindow ssm:DescribeActivations ssm:DescribeAssociation ssm:DescribeAssociationExecutionTargets ssm:DescribeAssociationExecutions ssm:DescribeAutomationExecutions ssm:DescribeAutomationStepExecutions ssm:DescribeAvailablePatches ssm:DescribeDocument ssm:DescribeDocumentParameters ssm:DescribeDocumentPermission ssm:DescribeEffectiveInstanceAssociations ssm:DescribeEffectivePatchesForPatchBaseline ssm:DescribeInstanceAssociationsStatus ssm:DescribeInstanceInformation

服务前缀	操作
	ssm:DescribeInstancePatchStates ssm:DescribeInstancePatchStatesForPatchGroup ssm:DescribeInstancePatches ssm:DescribeInstanceProperties ssm:DescribeInventoryDeletions ssm:DescribeMaintenanceWindowExecutionTaskInvocations ssm:DescribeMaintenanceWindowExecutionTasks ssm:DescribeMaintenanceWindowExecutions ssm:DescribeMaintenanceWindowSchedule ssm:DescribeMaintenanceWindowTargets ssm:DescribeMaintenanceWindowTasks ssm:DescribeMaintenanceWindows ssm:DescribeMaintenanceWindowsForTarget ssm:DescribeOpsItems ssm:DescribeParameters ssm:DescribePatchBaselines ssm:DescribePatchGroupState ssm:DescribePatchGroups ssm:DescribePatchProperties ssm:DescribeSessions ssm:DisassociateOpsItemRelatedItem

服务前缀	操作
	ssm:GetAccessToken ssm:GetAutomationExecution ssm:GetCalendarState ssm:GetCommandInvocation ssm:GetConnectionStatus ssm:GetDefaultPatchBaseline ssm:GetDeployablePatchSnapshotForInstance ssm:GetDocument ssm:GetExecutionPreview ssm:GetInventory ssm:GetInventorySchema ssm:GetMaintenanceWindow ssm:GetMaintenanceWindowExecution ssm:GetMaintenanceWindowExecutionTask ssm:GetMaintenanceWindowExecutionTaskInvocation ssm:GetMaintenanceWindowTask ssm:GetOpsItem ssm:GetOpsMetadata ssm:GetOpsSummary ssm:GetParameter ssm:GetParameterHistory

服务前缀	操作
	ssm:GetParameters ssm:GetParametersByPath ssm:GetPatchBaseline ssm:GetPatchBaselineForPatchGroup ssm:GetResourcePolicies ssm:GetServiceSetting ssm:LabelParameterVersion ssm:ListAssociationVersions ssm:ListAssociations ssm:ListCommandInvocations ssm:ListCommands ssm:ListComplianceItems ssm:ListComplianceSummaries ssm:ListDocumentMetadataHistory ssm:ListDocumentVersions ssm:ListDocuments ssm:ListInstanceAssociations ssm:ListInventoryEntries ssm:ListNodes ssm:ListNodesSummary ssm:ListOpsItemEvents

服务前缀	操作
	ssm:ListOpsItemRelatedItems ssm:ListOpsMetadata ssm:ListResourceComplianceSummaries ssm:ListResourceDataSync ssm:ModifyDocumentPermission ssm:PutComplianceItems ssm:PutInventory ssm:PutParameter ssm:PutResourcePolicy ssm:RegisterDefaultPatchBaseline ssm:RegisterManagedInstance ssm:RegisterPatchBaselineForPatchGroup ssm:RegisterTargetWithMaintenanceWindow ssm:RegisterTaskWithMaintenanceWindow ssm:ResetServiceSetting ssm:ResumeSession ssm:SendAutomationSignal ssm:SendCommand ssm:StartAssociationsOnce ssm:StartAutomationExecution ssm:StartChangeRequestExecution

服务前缀	操作
	ssm:StartSession ssm:StopAutomationExecution ssm:TerminateSession ssm:UnlabelParameterVersion ssm:UpdateAssociation ssm:UpdateAssociationStatus ssm:UpdateDocument ssm:UpdateDocumentDefaultVersion ssm:UpdateDocumentMetadata ssm:UpdateInstanceInformation ssm:UpdateMaintenanceWindow ssm:UpdateMaintenanceWindowTarget ssm:UpdateMaintenanceWindowTask ssm:UpdateManagedInstanceRole ssm:UpdateOpsItem ssm:UpdateOpsMetadata ssm:UpdatePatchBaseline ssm:UpdateResourceDataSync ssm:UpdateServiceSetting

服务前缀	操作
ssm-incidents	ssm-incidents:BatchGetIncidentFindings
	ssm-incidents:CreateReplicationSet
	ssm-incidents:CreateResponsePlan
	ssm-incidents:CreateTimelineEvent
	ssm-incidents:DeleteIncidentRecord
	ssm-incidents:DeleteReplicationSet
	ssm-incidents:DeleteResourcePolicy
	ssm-incidents:DeleteResponsePlan
	ssm-incidents:DeleteTimelineEvent
	ssm-incidents:GetIncidentRecord
	ssm-incidents:GetReplicationSet
	ssm-incidents:GetResourcePolicies
	ssm-incidents:GetResponsePlan
	ssm-incidents:GetTimelineEvent
	ssm-incidents:ListIncidentFindings
	ssm-incidents:ListIncidentRecords
	ssm-incidents:ListRelatedItems
	ssm-incidents:ListReplicationSets
	ssm-incidents:ListResponsePlans
	ssm-incidents:ListTimelineEvents
	ssm-incidents:PutResourcePolicy

服务前缀	操作
	ssm-incidents:StartIncident
	ssm-incidents:UpdateDeletionProtection
	ssm-incidents:UpdateIncidentRecord
	ssm-incidents:UpdateRelatedItems
	ssm-incidents:UpdateReplicationSet
	ssm-incidents:UpdateResponsePlan
	ssm-incidents:UpdateTimelineEvent

服务前缀	操作
ssm-sap	ssm-sap:BackupDatabase ssm-sap:DeleteResourcePermission ssm-sap:DeregisterApplication ssm-sap:GetApplication ssm-sap:GetComponent ssm-sap:GetConfigurationCheckOperation ssm-sap:GetDatabase ssm-sap:GetOperation ssm-sap:GetResourcePermission ssm-sap:ListApplications ssm-sap:ListComponents ssm-sap:ListConfigurationCheckDefinitions ssm-sap:ListConfigurationCheckOperations ssm-sap:ListDatabases ssm-sap:ListOperationEvents ssm-sap:ListOperations ssm-sap:ListSubCheckResults ssm-sap:ListSubCheckRuleResults ssm-sap:PutResourcePermission ssm-sap:RegisterApplication ssm-sap:RestoreDatabase

服务前缀	操作
	ssm-sap:StartApplication ssm-sap:StartApplicationRefresh ssm-sap:StartConfigurationChecks ssm-sap:StopApplication ssm-sap:UpdateApplicationSettings ssm-sap:UpdateHANABackupSettings

服务前缀	操作
states	states:CreateActivity
	states:CreateStateMachine
	states:CreateStateMachineAlias
	states:DeleteActivity
	states:DeleteStateMachine
	states:DeleteStateMachineAlias
	states:DeleteStateMachineVersion
	states:DescribeActivity
	states:DescribeExecution
	states:DescribeMapRun
	states:DescribeStateMachine
	states:DescribeStateMachineAlias
	states:DescribeStateMachineForExecution
	states:GetExecutionHistory
	states:ListActivities
	states:ListExecutions
	states:ListMapRuns
	states:ListStateMachineAliases
	states:ListStateMachineVersions
	states:ListStateMachines
	states:SendTaskFailure

服务前缀	操作
	states:SendTaskHeartbeat states:SendTaskSuccess states:StartExecution states:StopExecution states:UpdateMapRun states:UpdateStateMachine states:UpdateStateMachineAlias states:ValidateStateMachineDefinition
sts	sts:AssumeRole sts:AssumeRoleWithSAML sts:AssumeRoleWithWebIdentity sts:DecodeAuthorizationMessage sts:GetAccessKeyInfo sts:GetCallerIdentity sts:GetFederationToken sts:GetSessionToken sts:GetWebIdentityToken

服务前缀	操作
swf	swf:DeleteActivityType
	swf:DeleteWorkflowType
	swf:DeprecateActivityType
	swf:DeprecateDomain
	swf:DeprecateWorkflowType
	swf:DescribeActivityType
	swf:DescribeDomain
	swf:DescribeWorkflowType
	swf:ListActivityTypes
	swf:ListDomains
	swf:ListWorkflowTypes
	swf:RegisterActivityType
	swf:RegisterDomain
	swf:RegisterWorkflowType
	swf:UndeprecateActivityType
	swf:UndeprecateDomain
	swf:UndeprecateWorkflowType

服务前缀	操作
synthetics	synthetics:AssociateResource synthetics:CreateCanary synthetics:CreateGroup synthetics:DeleteCanary synthetics:DeleteGroup synthetics:DescribeCanaries synthetics:DescribeCanariesLastRun synthetics:DescribeRuntimeVersions synthetics:DisassociateResource synthetics:GetCanary synthetics:GetCanaryRuns synthetics:GetGroup synthetics:ListAssociatedGroups synthetics:ListGroupResources synthetics:ListGroups synthetics:StartCanary synthetics:StartCanaryDryRun synthetics:StopCanary synthetics:UpdateCanary

服务前缀	操作
标签	tag:DescribeReportCreation tag:GetComplianceSummary tag:GetResources tag:StartReportCreation

服务前缀	操作
textract	textract:AnalyzeDocument textract:AnalyzeExpense textract:AnalyzeID textract:CreateAdapter textract:CreateAdapterVersion textract:DeleteAdapter textract:DeleteAdapterVersion textract:DetectDocumentText textract:GetAdapter textract:GetAdapterVersion textract:GetDocumentAnalysis textract:GetDocumentTextDetection textract:GetExpenseAnalysis textract:GetLendingAnalysis textract:GetLendingAnalysisSummary textract:ListAdapterVersions textract:ListAdapters textract:StartDocumentAnalysis textract:StartDocumentTextDetection textract:StartExpenseAnalysis textract:StartLendingAnalysis

服务前缀	操作
	textract:UpdateAdapter

服务前缀	操作
timestream	timestream:CancelQuery timestream:CreateDatabase timestream:CreateScheduledQuery timestream:CreateTable timestream:DeleteDatabase timestream:DeleteScheduledQuery timestream:DeleteTable timestream:DescribeAccountSettings timestream:DescribeDatabase timestream:DescribeScheduledQuery timestream:DescribeTable timestream:ExecuteScheduledQuery timestream:ListBatchLoadTasks timestream:ListDatabases timestream:ListScheduledQueries timestream:ListTables timestream:PrepareQuery timestream:UpdateAccountSettings timestream:UpdateDatabase timestream:UpdateScheduledQuery timestream:UpdateTable

服务前缀	操作
tnb	tnb:CancelSolNetworkOperation tnb:CreateSolFunctionPackage tnb:CreateSolNetworkInstance tnb:CreateSolNetworkPackage tnb:DeleteSolFunctionPackage tnb:DeleteSolNetworkInstance tnb:DeleteSolNetworkPackage tnb:GetSolFunctionInstance tnb:GetSolFunctionPackage tnb:GetSolFunctionPackageContent tnb:GetSolFunctionPackageDescriptor tnb:GetSolNetworkInstance tnb:GetSolNetworkOperation tnb:GetSolNetworkPackage tnb:GetSolNetworkPackageContent tnb:GetSolNetworkPackageDescriptor tnb:InstantiateSolNetworkInstance tnb:ListSolFunctionInstances tnb:ListSolFunctionPackages tnb:ListSolNetworkInstances tnb:ListSolNetworkOperations

服务前缀	操作
	tnb:ListSolNetworkPackages tnb:PutSolFunctionPackageContent tnb:PutSolNetworkPackageContent tnb:TerminateSolNetworkInstance tnb:UpdateSolFunctionPackage tnb:UpdateSolNetworkInstance tnb:UpdateSolNetworkPackage tnb:ValidateSolFunctionPackageContent tnb:ValidateSolNetworkPackageContent

服务前缀	操作
transcribe	transcribe:CreateCallAnalyticsCategory transcribe:CreateLanguageModel transcribe:CreateMedicalVocabulary transcribe:CreateVocabulary transcribe:CreateVocabularyFilter transcribe:DeleteCallAnalyticsCategory transcribe:DeleteCallAnalyticsJob transcribe:DeleteLanguageModel transcribe:DeleteMedicalScribeJob transcribe:DeleteMedicalTranscriptionJob transcribe:DeleteMedicalVocabulary transcribe:DeleteTranscriptionJob transcribe:DeleteVocabulary transcribe:DeleteVocabularyFilter transcribe:DescribeLanguageModel transcribe:GetCallAnalyticsCategory transcribe:GetCallAnalyticsJob transcribe:GetMedicalScribeJob transcribe:GetMedicalTranscriptionJob transcribe:GetMedicalVocabulary transcribe:GetTranscriptionJob

服务前缀	操作
	transcribe:GetVocabulary
	transcribe:GetVocabularyFilter
	transcribe:ListCallAnalyticsCategories
	transcribe:ListCallAnalyticsJobs
	transcribe:ListLanguageModels
	transcribe:ListMedicalScribeJobs
	transcribe:ListMedicalTranscriptionJobs
	transcribe:ListMedicalVocabularies
	transcribe:ListTranscriptionJobs
	transcribe:ListVocabularies
	transcribe:ListVocabularyFilters
	transcribe:StartCallAnalyticsJob
	transcribe:StartCallAnalyticsStreamTranscription
	transcribe:StartCallAnalyticsStreamTranscriptionWebSocket
	transcribe:StartMedicalScribeJob
	transcribe:StartMedicalStreamTranscription
	transcribe:StartMedicalStreamTranscriptionWebSocket
	transcribe:StartMedicalTranscriptionJob
	transcribe:StartStreamTranscription
	transcribe:StartStreamTranscriptionWebSocket
	transcribe:StartTranscriptionJob

服务前缀	操作
	transcribe:UpdateCallAnalyticsCategory transcribe:UpdateMedicalVocabulary transcribe:UpdateVocabulary transcribe:UpdateVocabularyFilter

服务前缀	操作
转移	transfer:CreateAccess transfer:CreateAgreement transfer:CreateConnector transfer:CreateProfile transfer:CreateServer transfer:CreateUser transfer:CreateWebApp transfer:CreateWorkflow transfer:DeleteAccess transfer:DeleteAgreement transfer:DeleteCertificate transfer:DeleteConnector transfer:DeleteHostKey transfer:DeleteProfile transfer:DeleteServer transfer:DeleteSshPublicKey transfer:DeleteUser transfer:DeleteWebApp transfer:DeleteWebAppCustomization transfer:DeleteWorkflow transfer:DescribeAccess

服务前缀	操作
	transfer:DescribeAgreement transfer:DescribeCertificate transfer:DescribeConnector transfer:DescribeExecution transfer:DescribeHostKey transfer:DescribeProfile transfer:DescribeSecurityPolicy transfer:DescribeServer transfer:DescribeUser transfer:DescribeWebApp transfer:DescribeWebAppCustomization transfer:DescribeWorkflow transfer:ImportCertificate transfer:ImportHostKey transfer:ImportSshPublicKey transfer:ListAccesses transfer:ListCertificates transfer:ListConnectors transfer:ListExecutions transfer:ListFileTransferResults transfer:ListHostKeys

服务前缀	操作
	transfer:ListProfiles transfer:ListSecurityPolicies transfer:ListServers transfer:ListUsers transfer:ListWebApps transfer:ListWorkflows transfer:SendWorkflowStepState transfer:StartDirectoryListing transfer:StartFileTransfer transfer:StartRemoteDelete transfer:StartRemoteMove transfer:StartServer transfer:StopServer transfer:TestConnection transfer:TestIdentityProvider transfer:UpdateAccess transfer:UpdateAgreement transfer:UpdateCertificate transfer:UpdateConnector transfer:UpdateHostKey transfer:UpdateProfile

服务前缀	操作
	transfer:UpdateServer transfer:UpdateUser transfer:UpdateWebApp transfer:UpdateWebAppCustomization
翻译	translate:CreateParallelData translate:DeleteParallelData translate:DeleteTerminology translate:DescribeTextTranslationJob translate:GetParallelData translate:GetTerminology translate:ImportTerminology translate:ListLanguages translate:ListParallelData translate:ListTerminologies translate:ListTextTranslationJobs translate:StartTextTranslationJob translate:StopTextTranslationJob translate:TranslateDocument translate:TranslateText translate:UpdateParallelData

服务前缀	操作
voiceid	voiceid:AssociateFraudster voiceid:CreateDomain voiceid:CreateWatchlist voiceid:DeleteDomain voiceid:DeleteFraudster voiceid:DeleteSpeaker voiceid:DeleteWatchlist voiceid:DescribeDomain voiceid:DescribeFraudster voiceid:DescribeFraudsterRegistrationJob voiceid:DescribeSpeaker voiceid:DescribeSpeakerEnrollmentJob voiceid:DescribeWatchlist voiceid:DisassociateFraudster voiceid:EvaluateSession voiceid:ListDomains voiceid:ListFraudsterRegistrationJobs voiceid:ListFraudsters voiceid:ListSpeakerEnrollmentJobs voiceid:ListSpeakers voiceid:ListWatchlists

服务前缀	操作
	voiceid:OptOutSpeaker voiceid:StartFraudsterRegistrationJob voiceid:StartSpeakerEnrollmentJob voiceid:UpdateDomain voiceid:UpdateWatchlist

服务前缀	操作
vpc-lattice	vpc-lattice:CreateAccessLogSubscription
	vpc-lattice:CreateListener
	vpc-lattice:CreateResourceConfiguration
	vpc-lattice:CreateResourceGateway
	vpc-lattice:CreateRule
	vpc-lattice:CreateService
	vpc-lattice:CreateServiceNetwork
	vpc-lattice:CreateServiceNetworkResourceAssociation
	vpc-lattice:CreateServiceNetworkServiceAssociation
	vpc-lattice:CreateServiceNetworkVpcAssociation
	vpc-lattice:CreateTargetGroup
	vpc-lattice:DeleteAccessLogSubscription
	vpc-lattice:DeleteAuthPolicy
	vpc-lattice:DeleteDomainVerification
	vpc-lattice:DeleteListener
	vpc-lattice:DeleteResourceConfiguration
	vpc-lattice:DeleteResourceEndpointAssociation
	vpc-lattice:DeleteResourceGateway
	vpc-lattice:DeleteResourcePolicy
	vpc-lattice:DeleteRule
	vpc-lattice:DeleteService

服务前缀	操作
	vpc-lattice:DeleteServiceNetwork
	vpc-lattice:DeleteServiceNetworkResourceAssociation
	vpc-lattice:DeleteServiceNetworkServiceAssociation
	vpc-lattice:DeleteServiceNetworkVpcAssociation
	vpc-lattice:DeleteTargetGroup
	vpc-lattice:DeregisterTargets
	vpc-lattice:GetAccessLogSubscription
	vpc-lattice:GetAuthPolicy
	vpc-lattice:GetDomainVerification
	vpc-lattice:GetListener
	vpc-lattice:GetResourceConfiguration
	vpc-lattice:GetResourceGateway
	vpc-lattice:GetResourcePolicy
	vpc-lattice:GetRule
	vpc-lattice:GetService
	vpc-lattice:GetServiceNetwork
	vpc-lattice:GetServiceNetworkResourceAssociation
	vpc-lattice:GetServiceNetworkServiceAssociation
	vpc-lattice:GetServiceNetworkVpcAssociation
	vpc-lattice:GetTargetGroup
	vpc-lattice:ListAccessLogSubscriptions

服务前缀	操作
	vpc-lattice:ListDomainVerifications
	vpc-lattice:ListListeners
	vpc-lattice:ListResourceConfigurations
	vpc-lattice:ListResourceEndpointAssociations
	vpc-lattice:ListResourceGateways
	vpc-lattice:ListRules
	vpc-lattice:ListServiceNetworkResourceAssociations
	vpc-lattice:ListServiceNetworkServiceAssociations
	vpc-lattice:ListServiceNetworkVpcAssociations
	vpc-lattice:ListServiceNetworkVpcEndpointAssociations
	vpc-lattice:ListServiceNetworks
	vpc-lattice:ListServices
	vpc-lattice:ListTargetGroups
	vpc-lattice:ListTargets
	vpc-lattice:PutAuthPolicy
	vpc-lattice:PutResourcePolicy
	vpc-lattice:RegisterTargets
	vpc-lattice:StartDomainVerification
	vpc-lattice:UpdateAccessLogSubscription
	vpc-lattice:UpdateListener
	vpc-lattice:UpdateResourceConfiguration

服务前缀	操作
	vpc-lattice:UpdateResourceGateway
	vpc-lattice:UpdateRule
	vpc-lattice:UpdateService
	vpc-lattice:UpdateServiceNetwork
	vpc-lattice:UpdateServiceNetworkVpcAssociation
	vpc-lattice:UpdateTargetGroup

服务前缀	操作
wafv2	wafv2:AssociateWebACL wafv2:CheckCapacity wafv2:CreateAPIKey wafv2:CreateIPSet wafv2:CreateRegexPatternSet wafv2:CreateRuleGroup wafv2:CreateWebACL wafv2:DeleteAPIKey wafv2:DeleteFirewallManagerRuleGroups wafv2:DeleteIPSet wafv2:DeleteLoggingConfiguration wafv2:DeletePermissionPolicy wafv2:DeleteRegexPatternSet wafv2:DeleteRuleGroup wafv2:DeleteWebACL wafv2:DescribeAllManagedProducts wafv2:DescribeManagedProductsByVendor wafv2:DescribeManagedRuleGroup wafv2:DisassociateWebACL wafv2:GenerateMobileSdkReleaseUrl wafv2:GetDecryptedAPIKey

服务前缀	操作
	wafv2:GetIPSet wafv2:GetLoggingConfiguration wafv2:GetManagedRuleSet wafv2:GetMobileSdkRelease wafv2:GetRateBasedStatementManagedKeys wafv2:GetRegexPatternSet wafv2:GetRuleGroup wafv2:GetSampledRequests wafv2:GetWebACLForResource wafv2:ListAPIKeys wafv2:ListAvailableManagedRuleGroupVersions wafv2:ListAvailableManagedRuleGroups wafv2:ListIPSets wafv2:ListLoggingConfigurations wafv2:ListManagedRuleSets wafv2:ListMobileSdkReleases wafv2:ListRegexPatternSets wafv2:ListResourcesForWebACL wafv2:ListRuleGroups wafv2:ListWebACLs wafv2:PutLoggingConfiguration

服务前缀	操作
	wafv2:PutManagedRuleSetVersions wafv2:UpdateIPSet wafv2:UpdateManagedRuleSetVersionExpiryDate wafv2:UpdateRegexPatternSet wafv2:UpdateRuleGroup wafv2:UpdateWebACL

服务前缀	操作
wellarchitected	wellarchitected:AssociateLenses
	wellarchitected:AssociateProfiles
	wellarchitected:CreateLensShare
	wellarchitected:CreateLensVersion
	wellarchitected:CreateMilestone
	wellarchitected:CreateProfile
	wellarchitected:CreateProfileShare
	wellarchitected:CreateReviewTemplate
	wellarchitected:CreateWorkload
	wellarchitected:CreateWorkloadShare
	wellarchitected:DeleteLens
	wellarchitected:DeleteLensShare
	wellarchitected:DeleteProfile
	wellarchitected:DeleteProfileShare
	wellarchitected:DeleteReviewTemplate
	wellarchitected:DeleteTemplateShare
	wellarchitected:DeleteWorkload
	wellarchitected:DeleteWorkloadShare
	wellarchitected:DisassociateLenses
	wellarchitected:DisassociateProfiles
	wellarchitected:ExportLens

服务前缀	操作
	wellarchitected:GetAnswer
	wellarchitected:GetConsolidatedReport
	wellarchitected:GetGlobalSettings
	wellarchitected:GetLens
	wellarchitected:GetLensReview
	wellarchitected:GetLensReviewReport
	wellarchitected:GetLensVersionDifference
	wellarchitected:GetMilestone
	wellarchitected:GetProfile
	wellarchitected:GetProfileTemplate
	wellarchitected:GetReviewTemplate
	wellarchitected:GetReviewTemplateAnswer
	wellarchitected:GetReviewTemplateLensReview
	wellarchitected:GetWorkload
	wellarchitected:ImportLens
	wellarchitected:ListAnswers
	wellarchitected:ListCheckDetails
	wellarchitected:ListCheckSummaries
	wellarchitected:ListLensReviewImprovements
	wellarchitected:ListLensReviews
	wellarchitected:ListLensShares

服务前缀	操作
	wellarchitected:ListLenses
	wellarchitected:ListMilestones
	wellarchitected:ListNotifications
	wellarchitected:ListProfileNotifications
	wellarchitected:ListProfileShares
	wellarchitected:ListProfiles
	wellarchitected:ListReviewTemplateAnswers
	wellarchitected:ListReviewTemplates
	wellarchitected:ListShareInvitations
	wellarchitected:ListTemplateShares
	wellarchitected:ListWorkloadShares
	wellarchitected:ListWorkloads
	wellarchitected:UpdateAnswer
	wellarchitected:UpdateGlobalSettings
	wellarchitected:UpdateIntegration
	wellarchitected:UpdateLensReview
	wellarchitected:UpdateProfile
	wellarchitected:UpdateReviewTemplate
	wellarchitected:UpdateReviewTemplateLensReview
	wellarchitected:UpdateShareInvitation
	wellarchitected:UpdateWorkload

服务前缀	操作
	wellarchitected:UpdateWorkloadShare wellarchitected:UpgradeLensReview wellarchitected:UpgradeProfileVersion wellarchitected:UpgradeReviewTemplateLensReview

服务前缀	操作
wisdom	wisdom:CreateAssistant wisdom:CreateAssistantAssociation wisdom:CreateContent wisdom:CreateKnowledgeBase wisdom:CreateQuickResponse wisdom:CreateSession wisdom:DeleteAssistant wisdom:DeleteAssistantAssociation wisdom:DeleteContent wisdom:DeleteImportJob wisdom:DeleteKnowledgeBase wisdom:DeleteQuickResponse wisdom:GetAssistant wisdom:GetAssistantAssociation wisdom:GetContent wisdom:GetContentAssociation wisdom:GetContentSummary wisdom:GetImportJob wisdom:GetKnowledgeBase wisdom:GetRecommendations wisdom:GetSession

服务前缀	操作
	wisdom:ListAssistantAssociations wisdom:ListAssistants wisdom:ListContentAssociations wisdom:ListContents wisdom:ListImportJobs wisdom:ListKnowledgeBases wisdom:ListQuickResponses wisdom:NotifyRecommendationsReceived wisdom:QueryAssistant wisdom:RemoveKnowledgeBaseTemplateUri wisdom:SearchContent wisdom:SearchQuickResponses wisdom:SearchSessions wisdom:StartContentUpload wisdom:StartImportJob wisdom:UpdateContent wisdom:UpdateKnowledgeBaseTemplateUri wisdom:UpdateQuickResponse wisdom:UpdateSession

服务前缀	操作
worklink	worklink:AssociateDomain worklink:AssociateWebsiteAuthorizationProvider worklink:AssociateWebsiteCertificateAuthority worklink:CreateFleet worklink:DeleteFleet worklink:DescribeAuditStreamConfiguration worklink:DescribeCompanyNetworkConfiguration worklink:DescribeDevice worklink:DescribeDevicePolicyConfiguration worklink:DescribeDomain worklink:DescribeFleetMetadata worklink:DescribeIdentityProviderConfiguration worklink:DescribeWebsiteCertificateAuthority worklink:DisassociateDomain worklink:DisassociateWebsiteAuthorizationProvider worklink:DisassociateWebsiteCertificateAuthority worklink:ListDevices worklink:ListDomains worklink:ListFleets worklink:ListWebsiteAuthorizationProviders worklink:ListWebsiteCertificateAuthorities

服务前缀	操作
	worklink:RestoreDomainAccess
	worklink:RevokeDomainAccess
	worklink:SignOutUser
	worklink:UpdateAuditStreamConfiguration
	worklink:UpdateCompanyNetworkConfiguration
	worklink:UpdateDevicePolicyConfiguration
	worklink:UpdateDomainMetadata
	worklink:UpdateFleetMetadata
	worklink:UpdateIdentityProviderConfiguration

服务前缀	操作
工作区	workspaces:AcceptAccountLinkInvitation workspaces:AssociateConnectionAlias workspaces:AssociateIpGroups workspaces:AssociateWorkspaceApplication workspaces:CopyWorkspaceImage workspaces:CreateAccountLinkInvitation workspaces:CreateConnectClientAddIn workspaces:CreateConnectionAlias workspaces:CreateIpGroup workspaces:CreateStandbyWorkspaces workspaces:CreateUpdatedWorkspaceImage workspaces:CreateWorkspaceBundle workspaces:CreateWorkspaceImage workspaces:CreateWorkspaces workspaces:CreateWorkspacesPool workspaces:DeleteAccountLinkInvitation workspaces:DeleteClientBranding workspaces:DeleteConnectClientAddIn workspaces:DeleteConnectionAlias workspaces:DeleteIpGroup workspaces:DeleteWorkspaceBundle

服务前缀	操作
	workspaces:DeleteWorkspaceImage workspaces:DeployWorkspaceApplications workspaces:DeregisterWorkspaceDirectory workspaces:DescribeAccount workspaces:DescribeAccountModifications workspaces:DescribeApplicationAssociations workspaces:DescribeApplications workspaces:DescribeBundleAssociations workspaces:DescribeClientBranding workspaces:DescribeClientProperties workspaces:DescribeConnectClientAddIns workspaces:DescribeConnectionAliasPermissions workspaces:DescribeConnectionAliases workspaces:DescribeCustomWorkspaceImageImport workspaces:DescribeImageAssociations workspaces:DescribeIpGroups workspaces:DescribeWorkspaceAssociations workspaces:DescribeWorkspaceBundles workspaces:DescribeWorkspaceDirectories workspaces:DescribeWorkspaceImagePermissions workspaces:DescribeWorkspaceSnapshots

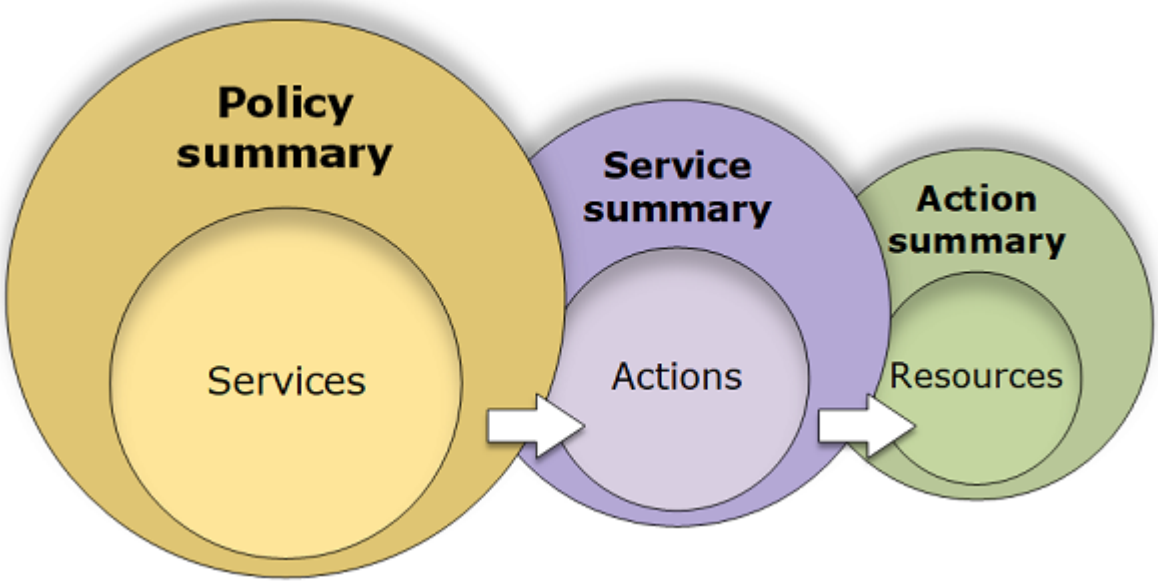
服务前缀	操作
	workspaces:DescribeWorkspaces workspaces:DescribeWorkspacesConnectionStatus workspaces:DescribeWorkspacesPoolSessions workspaces:DescribeWorkspacesPools workspaces:DisassociateConnectionAlias workspaces:DisassociateIpGroups workspaces:DisassociateWorkspaceApplication workspaces:GetAccountLink workspaces:ImportClientBranding workspaces:ImportWorkspaceImage workspaces:ListAccountLinks workspaces:ListAvailableManagementCidrRanges workspaces:MigrateWorkspace workspaces:ModifyAccount workspaces:ModifyCertificateBasedAuthProperties workspaces:ModifyClientProperties workspaces:ModifyEndpointEncryptionMode workspaces:ModifySamlProperties workspaces:ModifySelfservicePermissions workspaces:ModifyStreamingProperties workspaces:ModifyWorkspaceAccessProperties

服务前缀	操作
	workspaces:ModifyWorkspaceCreationProperties workspaces:ModifyWorkspaceProperties workspaces:ModifyWorkspaceState workspaces:RebootWorkspaces workspaces:RebuildWorkspaces workspaces:RegisterWorkspaceDirectory workspaces:RejectAccountLinkInvitation workspaces:RestoreWorkspace workspaces:StartWorkspaces workspaces:StartWorkspacesPool workspaces:StopWorkspaces workspaces:StopWorkspacesPool workspaces:TerminateWorkspaces workspaces:TerminateWorkspacesPool workspaces:TerminateWorkspacesPoolSession workspaces:UpdateConnectClientAddIn workspaces:UpdateConnectionAliasPermission workspaces:UpdateWorkspaceBundle workspaces:UpdateWorkspaceImagePermission workspaces:UpdateWorkspacesPool

服务前缀	操作
xray	xray:CreateGroup xray:CreateSamplingRule xray:DeleteGroup xray:DeleteResourcePolicy xray:DeleteSamplingRule xray:GetEncryptionConfig xray:GetGroup xray:GetGroups xray:GetInsight xray:GetInsightEvents xray:GetInsightImpactGraph xray:GetInsightSummaries xray:GetSamplingRules xray:ListResourcePolicies xray:PutEncryptionConfig xray:PutResourcePolicy xray:UpdateGroup xray:UpdateSamplingRule

策略摘要

IAM 控制台中提供了策略摘要表，这些表总结了策略中对每个服务允许或拒绝的访问级别、资源和条件。策略在三个表中概括：策略摘要、服务摘要和操作摘要。策略摘要表包含服务列表。选择其中的服务可查看服务摘要。该摘要表包含所选服务的操作和关联权限的列表。您可以选择该表中的操作以查看操作摘要。该表包含所选操作的资源和条件列表。



您可以在用户或角色页上查看附加到该用户的所有策略 (托管和内联) 的策略摘要。可在 Policies 页面上查看所有托管策略的摘要。托管策略包括 Amazon 托管策略、Amazon 托管工作职能策略和客户托管策略。您可以在 Policies (策略) 页面上查看这些策略的摘要，无论它们是附加到用户还是其他 IAM 身份。

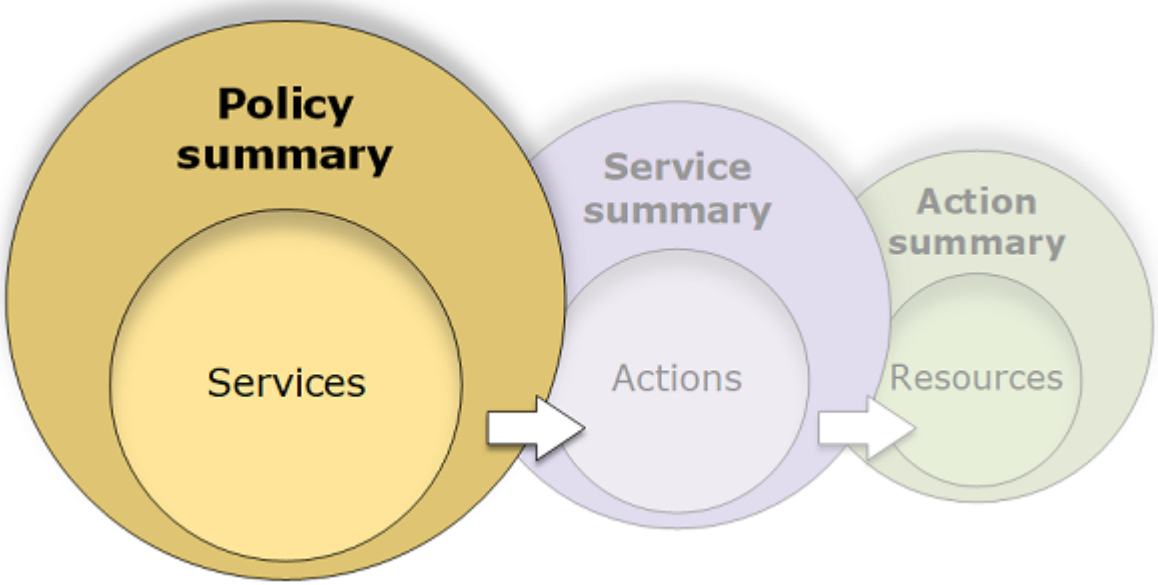
您可以使用策略摘要中的信息了解您的策略允许或拒绝的权限。策略摘要可帮助您排查故障并修复未提供您期望的权限的策略。

主题

- 策略摘要（服务列表）
- 策略摘要中的访问级别
- 服务摘要（操作列表）
- 操作摘要（资源列表）
- 策略摘要示例

策略摘要（服务列表）

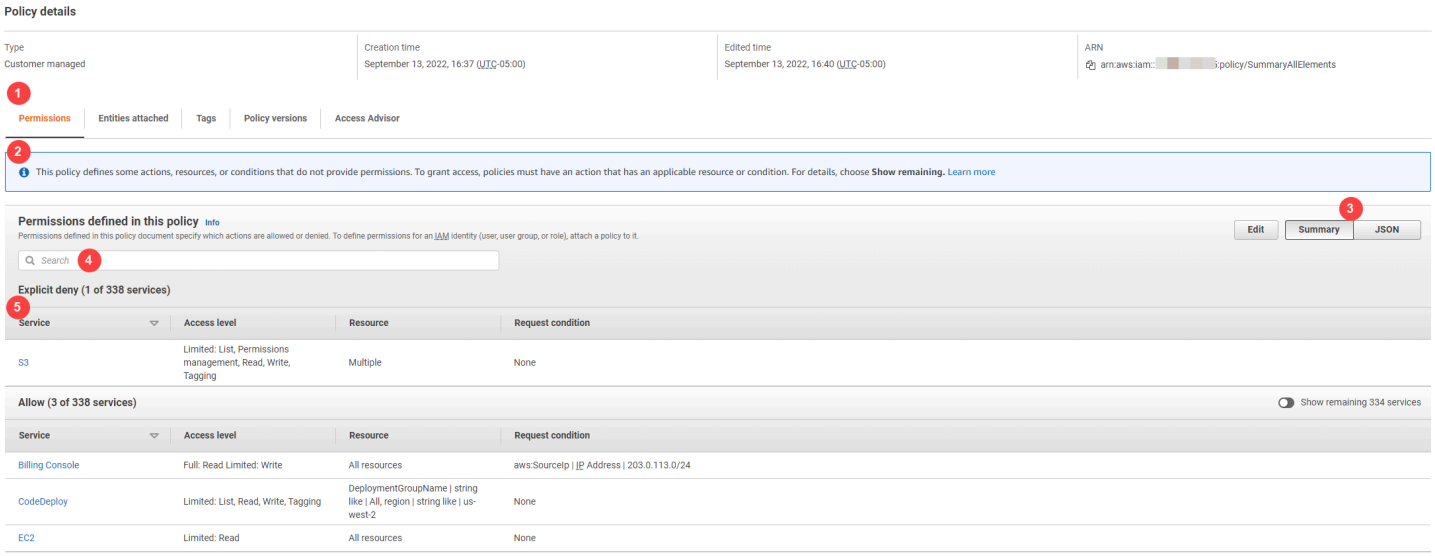
策略在三个表中概括：策略摘要、服务摘要和操作摘要。策略摘要表包括由所选策略定义的服务列表和权限摘要。



策略摘要表分为 Uncategorized services、Explicit deny 和 Allow 几部分。如果策略包含 IAM 无法识别的服务，则该服务将包含在表的 Uncategorized services（未分类服务）部分中。如果 IAM 能够识别服务，则该服务将包含在表的 Explicit deny（显式拒绝）或 Allow（允许）部分下，具体取决于策略作用的结果（`Deny` 或 `Allow`）。

了解策略摘要的元素

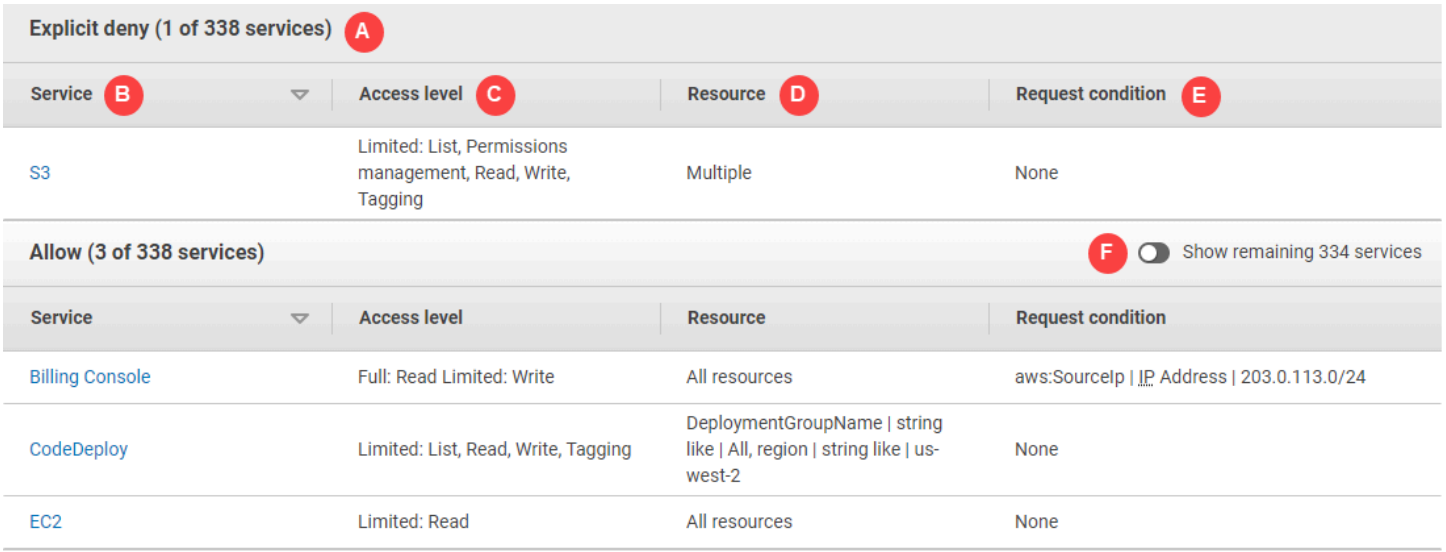
在以下策略详细信息页面示例中，SummaryAllElements 策略是直接附加到用户的管理型策略（客户管理型策略）。此策略已展开，显示了策略摘要。



在上图中，策略摘要在策略页面内可见：

1. 权限选项卡包括策略中定义的权限。
2. 如果策略未向策略中定义的所有操作、资源和条件授予权限，则将在页面顶部显示警告或错误横幅。策略摘要中包含有关问题的详细信息。要了解策略摘要如何帮助您了解策略授予的权限并进行相关问题排查，请参阅the section called “我的策略未授予预期权限”。
3. 使用摘要和 JSON 按钮可在策略摘要和 JSON 策略文档之间切换。
4. 使用搜索框可减少服务列表，查找特定服务。
5. 展开的视图显示了 SummaryAllElements 策略的更多详细信息。

下面的策略摘要表图像显示在策略详细信息页面上已展开的 SummaryAllElements 策略。



在上图中，策略摘要在策略页面内可见：

A. 对于 IAM 识别的那些服务，它根据策略是允许还是显式拒绝使用该服务来安排服务。在此示例中，策略包含用于 Amazon S3 服务的 `Deny` 语句和用于账单、CodeDeploy 和 Amazon EC2 服务的 `Allow` 语句。

B. Service（服务）- 此列将列出在策略内定义的服务并提供每项服务的详细信息。策略摘要表中的每个服务名称都是指向服务摘要 表的一个链接，服务摘要（操作列表）中对其进行了说明。在此示例中，为 Amazon S3、账单、CodeDeploy 和 Amazon EC2 服务定义了权限。

C. 访问级别 – 此列指出每个访问级别（`List`、`Read`、`Write`、`Permission Management` 和 `Tagging`）中的操作是具有策略中定义的 `Full` 还是 `Limited` 权限。有关访问权限级别摘要的更多详细信息和示例，请参阅策略摘要中的访问级别。

- Full access（完全访问权限）- 此条目指示服务对该服务可用的全部四个访问权限级别中的所有操作都拥有访问权限。
- 如果该项不包含 Full access，则服务可以访问部分但不是全部用于该服务的操作。然后，通过对各个访问级别分类（`List`、`Read`、`Write`、`Permission Management` 和 `Tagging`）的说明来定义访问权限：

  Full：策略提供对列出的每个访问级别分类中的所有操作的访问权限。在此示例中，策略提供对所有账单 `Read` 操作的访问权限。

  Limited：策略提供对所列每个访问级别分类内的一个或多个但不是全部操作的访问权限。在此示例中，策略提供对部分账单 `Write` 操作的访问权限。

D. Resource（资源）- 此列显示策略为每项服务指定的资源。

- Multiple（多个）- 策略包含服务内多个但不是全部资源。在此示例中，显式拒绝对多个 Amazon S3 资源的访问权限。
- 所有资源 – 策略是为服务内的所有资源定义的。在此示例中，策略允许对所有账单资源执行列出的操作。
- Resource text - 该策略包含服务内的一个资源。在此示例中，只允许对 `DeploymentGroupName` CodeDeploy 资源执行列出的操作。根据服务提供给 IAM 的信息，您可能会看到 ARN 或定义的资源类型。

> **Note**
>
> 此列可以包括来自不同服务的资源。如果包含资源的策略语句不包括来自同一服务的操作和资源，则您的策略将包括不匹配的资源。在创建策略或在策略摘要中查看策略时，IAM

不会就不匹配的资源向您发出警告。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。为了更好地了解您的策略，请始终使用策略模拟器进行测试。

E. Request condition（请求操作）- 此列指示与资源关联的服务或操作是否受条件约束。

- None（无）- 策略对服务不包含任何条件。在此示例中，没有条件适用于 Amazon S3 服务中拒绝的操作。
- Condition text - 策略对服务包含一个条件。在此示例中，仅当源的 IP 地址与 `203.0.113.0/24` 匹配时，才允许执行列出的账单操作。
- Multiple（多个）- 策略对服务包含多个条件。要查看策略多个条件中的每一个条件，请选择 JSON 查看策略文档。

F. 显示剩余服务 – 切换此按钮可展开表以包含策略未定义的服务。这些服务在该策略中被隐式拒绝 (或默认拒绝)。但是，另一策略中的语句可能仍然允许或显式拒绝使用该服务。策略摘要汇总了单个策略的权限。要了解 Amazon 服务如何决定是允许还是拒绝给定的请求，请参阅策略评估逻辑。

当策略或策略中的元素未授予权限时，IAM 在策略摘要中提供额外的警告和信息。下面的策略摘要表显示了在 SummaryAllElements 策略详细信息页面上展开的显示剩余的服务，并附带可能的警告。

Explicit deny (1 of 338 services)

Service	Access level	Resource a	Request condition b
S3	Limited: List, Permissions management, Read, Write, Tagging	c Multiple| ⚠ One or more actions do not have an applicable resource.	None

Allow (3 of 338 services) Show remaining 334 services

Service	Access level	Resource	Request condition
Billing Console	Full: Read Limited: Write	All resources	aws:SourceIp | IP Address | 203.0.113.0/24
CodeCommit	None	d ⚠ No resources are defined.	None
CodeDeploy	Limited: List, Read, Write, Tagging	e DeploymentGroupName | string like | All, region | string like | us-west-2| ⚠ One or more actions do not have an applicable resource.	None
EC2	Limited: Read	All resources	None
S3	None	None| ⚠ One or more actions do not have an applicable resource.	f None | ⚠ One or more conditions do not have an applicable action.

在上图中，您可以看到包含没有权限的既定操作、资源或条件的所有服务：

a. Resource warnings（资源警告）- 对于没有为所有包含的操作或资源提供权限的服务，您将在表的 Resource（资源）列中看到以下警告之一：

-

  No resources are defined. (未定义任何资源。) - 这意味着服务具有定义的操作，但策略不包含支持的资源。

-

  One or more actions do not have an applicable resource. (一个或多个操作没有适用的资源。) - 这意味着该服务具有定义的操作，但其中的一些操作没有支持的资源。

-

  One or more resources do not have an applicable action. (一个或多个资源没有适用的操作。) - 这意味着该服务具有定义的资源，但其中的一些资源没有支持的操作。

  如果服务同时包含没有适用资源的操作和有适用资源的资源，则显示警告：一个或多个资源没有适用的操作。这是因为：当您查看服务的服务摘要时，不会显示不适用于任何操作的资源。对于 `ListAllMyBuckets` 操作，该策略包含最后一条警告，因为该操作不支持资源级权限，也不支持 `s3:x-amz-acl` 条件键。如果修复了资源问题或条件问题，详细警告中将显示剩余问题。

b. Request condition warnings（请求条件警告）- 对于没有为所有包含的条件提供权限的服务，您将在表的 Request condition（请求条件）列中看到以下警告之一：

-

  One or more actions do not have an applicable condition. (一个或多个操作没有适用的条件。) - 这意味着服务具有定义的操作，但其中的一些操作没有支持的条件。

-

  One or more conditions do not have an applicable action. (一个或多个条件没有适用的操作。) - 这意味着服务定义了一些条件，但其中的一些条件没有支持的操作。

c. Multiple |

One or more actions do not have an applicable resource. (一个或多个操作没有适用的资源。) - Amazon S3 的 `Deny` 语句包含多个资源。它还包含多个操作，但其中一些操作支持资源，一些不支持。要查看该策略，请参阅 the section called “SummaryAllElements JSON 策略文档”。在这种情况下，策略包含所有 Amazon S3 操作，只拒绝可在存储桶或存储桶对象上执行的操作。

d. No resources are

defined

服务具有定义的操作，但策略中不包含支持的资源，因此服务不提供任何权限。在这种情况下，策略包含 CodeCommit 操作但不包含 CodeCommit 资源。

e. DeploymentGroupName | 字符串示例 | All, region | 字符串示例 | us-west-2 |

⚠

一个或多个资源没有适用的操作。– 服务具有定义的操作和至少一个没有支持资源的操作。

f. 无 |

⚠

一个或多个条件没有适用的操作。– 服务具有至少一个没有支持操作的条件键。

SummaryAllElements JSON 策略文档

SummaryAllElements 策略不适用于在账户中定义权限。包含它的目的在于演示您在查看策略摘要时可能会遇到的错误和警告。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "billing:Get*",
                "payments:List*",
                "payments:Update*",
                "account:Get*",
                "account:List*",
                "cur:GetUsage*"
            ],
            "Resource": [
                "*"
            ],
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action": [
                "s3:*"
```

```
            ],
            "Resource": [
                "arn:aws:s3:::customer",
                "arn:aws:s3:::customer/*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshots"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "codedploy:*",
                "codecommit:*"
            ],
            "Resource": [
                "arn:aws:codedeploy:us-west-2:123456789012:deploymentgroup:*",
                "arn:aws:codebuild:us-east-1:123456789012:project/my-demo-
project"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListAllMyBuckets",
                "s3:GetObject",
                "s3:DeletObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*",
                "arn:aws:autoscling:us-east-2:123456789012:autoscalgrp"
            ],
            "Condition": {
                "StringEquals": {
                    "s3:x-amz-acl": [
```

```
                    "public-read"
                ],
                "s3:prefix": [
                    "custom",
                    "other"
                ]
            }
        }
    }
  ]
}
```

查看策略摘要

可以查看附加到 IAM 用户或角色的任何策略的策略摘要。对于托管式策略，可以在策略页面上查看策略摘要。如果您的策略不包含策略摘要，请参阅缺少策略摘要了解原因。

从策略页面查看策略摘要

可以在 Policies 页面上查看托管策略的策略摘要。

从 Policies 页面查看策略摘要

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要查看的策略的名称。
4. 在策略的策略详细信息页面上，查看权限选项卡以查看策略摘要。

查看附加到用户的策略的策略摘要

可以查看附加到 IAM 用户的任何策略的策略摘要。

查看附加到用户的策略的摘要

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 从导航窗格中选择 Users。
3. 在用户列表中，选择要查看其策略的用户的名称。

4. 在用户的 Summary 页面上，查看 Permissions 选项卡，以查看直接附加到用户或从组附加到用户的策略列表。
5. 在用户的策略表中，展开要查看的策略的行。

查看附加到角色的策略的策略摘要

可以查看附加到角色的任何策略的策略摘要。

查看附加到角色的策略的摘要

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 在角色列表中，选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上，查看 Permissions 选项卡，以查看附加到角色的策略列表。
5. 在角色的策略表中，展开要查看的策略的行。

编辑策略以纠正警告

在查看策略摘要时，您可能会发现拼写错误，或者注意到策略未提供所需的权限。您无法直接编辑策略摘要。不过，您可以使用可视化策略编辑器编辑客户管理的策略，该编辑器捕获策略摘要报告的很多相同的错误和警告。然后，您可以在策略摘要中查看更改以确认纠正了所有问题。要了解如何编辑内联策略，请参阅the section called “编辑 IAM 策略”。您无法编辑 Amazon 托管策略。

可以使用可视化选项编辑策略摘要的策略。

使用可视化选项为您的策略摘要编辑策略

1. 按照上述步骤中的说明打开策略摘要。
2. 选择编辑。

   如果您已打开用户页面，请选择编辑附加到该用户的客户托管策略，之后您会被重定向到策略页面。您只能在 Policies 页面上编辑客户托管策略。
3. 请选择可视化选项以查看您的策略的可编辑可视化形式。IAM 可能会调整您的策略结构以针对可视化编辑器进行优化，并使您更轻松地查找和解决任何问题。页面上的警告和错误消息可以指导您解决策略中的任何问题。有关 IAM 如何调整策略结构的更多信息，请参阅 调整策略结构。

4. 编辑您的策略，然后选择下一步以查看策略摘要中是否反映了更改。如果仍出现问题，请选择上一步以返回到编辑屏幕。
5. 选择保存更改以保存您的更改。

可以使用 JSON 选项编辑策略摘要的策略。

使用 JSON 选项编辑策略摘要的策略

1. 按照上述步骤中的说明打开策略摘要。
2. 您可以使用摘要和 JSON 按钮比较策略摘要和 JSON 策略文档。您可以使用该信息来确定要更改策略文档中的哪些行。
3. 选择编辑，然后选择 JSON 选项以编辑 JSON 策略文档。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器选项中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

如果您已打开用户页面，请选择编辑附加到该用户的客户托管策略，之后您会被重定向到策略页面。您只能在 Policies 页面上编辑客户托管策略。

4. 编辑策略。解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。如果仍出现问题，请选择上一步以返回到编辑屏幕。
5. 选择保存更改以保存您的更改。

策略摘要中的访问级别

Amazon 访问级别摘要

策略摘要中包括一个访问级别摘要，用于描述为策略中提及的每项服务定义的操作权限。要了解策略摘要，请参阅策略摘要。访问级别摘要指出在策略中为每个访问级别中的操作（`List`、`Read`、`Tagging`、`Write` 和 `Permissions management`）定义的是 `Full` 还是 `Limited` 权限。要查看分配给服务中每个操作的访问级别分类，请参阅 Amazon 服务的操作、资源和条件键。

下面的示例介绍策略为给定服务提供的访问权限。有关完整 JSON 策略文档及其相关摘要的示例，请参阅策略摘要示例。

服务	访问级别	此策略提供以下访问权限
IAM	完全访问	对 IAM 服务内所有操作的访问权限。
CloudWatch	Full: List	对 List 访问级别中所有 CloudWatch 操作的访问权限，但无权访问 Read、Write或 Permissions management 访问级别分类的操作。
Data Pipeline	Limited: List, Read	对于 List 和 Read 访问级别中至少一项但不是全部 Amazon Data Pipeline 操作（但不是 Write 或 Permissions management 操作）的访问权限。
EC2	Full: List, Read Limited: Write	对所有 Amazon EC2List 和 Read 操作的访问权限以及对至少一项但不是全部 Amazon EC2 Write 操作的访问权限，但不具有对于 Permissions management 访问级别分类中的操作的访问权限。
S3	Limited：Read、Write、Permissions management	访问至少一个但并非全部 Amazon S3 Read、Write 和 Permissions management 操作。
codedploy	（空）	未知访问权限，因为 IAM 无法识别此服务。
API Gateway	无	策略中未定义任何访问权限。
CodeBuild	⚠ 未定义任何操作。	没有为服务定义任何操作，因而无法访问。要了解该问题和进行问题排查，请参阅the section called “我的策略未授予预期权限”。

在策略摘要中，完全访问权限表示策略提供对服务内所有操作的访问权限。提供对服务内的部分但不是全部操作的访问权限的策略将根据访问级别分类进一步分组。这由下面的其中一个访问级别分组来指示：

- Full：策略提供对指定访问级别分类中所有操作的访问权限。
- Limited：策略提供对指定访问级别分类内的一个或多个但不是全部操作的访问权限。
- None：策略未提供任何访问权限。
- （空）：IAM 无法识别该服务。如果服务名称包含拼写错误，则该策略不允许访问该服务。如果服务名称正确，则服务可能不支持策略摘要或可能正处于预览状态。在这种情况下，策略可能会提供访问权限，但访问权限可能不会显示在策略摘要中。要为公开提供 (GA) 服务请求策略摘要支持，请参阅服务不支持 IAM policy 摘要。

包括对操作的有限（部分）访问权限的访问级别摘要使用 Amazon 服务级别分类 `List`、`Read`、`Tagging`、`Write` 或 `Permissions management` 进行分组。

Amazon 访问级别

Amazon 为服务中的操作定义以下访问级别分类：

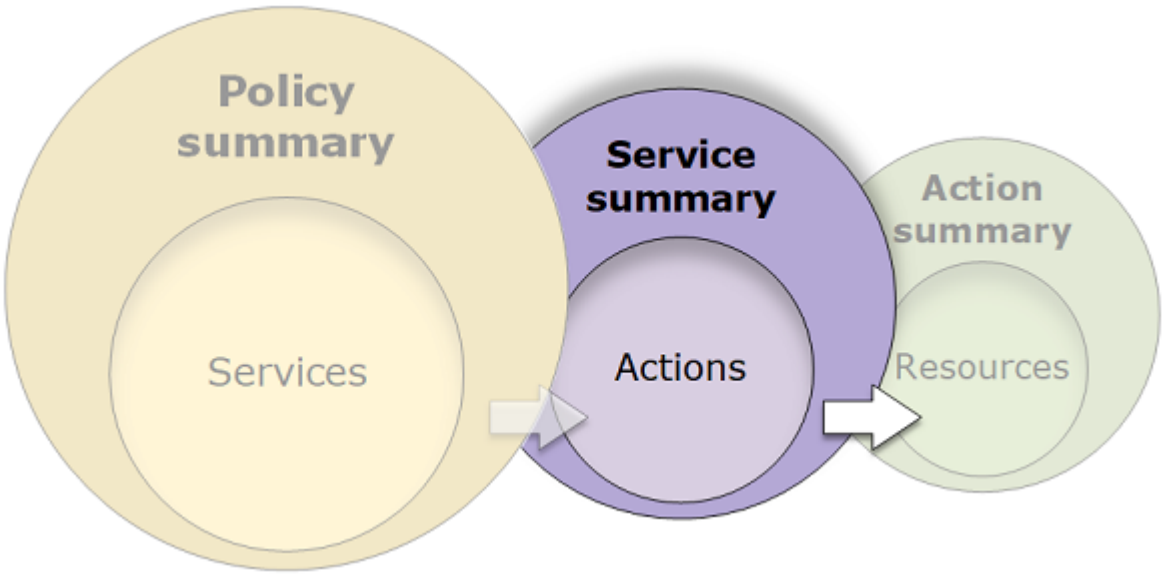
- List (列出)：列出服务内的资源以确定某个对象是否存在的权限。此访问权限级别的操作可以列出对象，但是看不到资源的内容。例如，Amazon S3 操作 `ListBucket` 具有 List（列出）访问级别。
- Read (读取)：读取服务中资源的内容和属性但不对其进行编辑的权限。例如，Amazon S3 操作 `GetObject` 和 `GetBucketLocation` 具有 Read（读取）访问权限级别。
- 标记：执行仅更改资源标签状态的操作的权限。例如，IAM 操作 `TagRole` 和 `UntagRole` 具有标记访问级别，因为它们仅允许标记或取消标记角色。不过，`CreateRole` 操作允许在创建角色时标记该角色资源。由于该操作并非仅添加标签，因此，它具有 `Write` 访问级别。
- Write (写入)：在服务中创建、删除或修改资源的权限。例如，Amazon S3 操作 `CreateBucket`、`DeleteBucket` 和 `PutObject` 具有写入访问级别。`Write` 操作可能还允许修改资源标签。不过，仅允许更改标签的操作具有 `Tagging` 访问级别。
- 权限管理：权限管理是指控制 Amazon Web Services 服务 内访问的操作，包括 IAM 和非 IAM 身份权限，但不包括安全组等网络级访问控制。例如，大多数 IAM 和 Amazon Organizations 操作以及 Amazon S3 操作 `PutBucketPolicy` 和 `DeleteBucketPolicy` 具有权限管理访问级别。

> **提示**
>
> 要提高您的 Amazon Web Services 账户 的安全性，请限制或定期监控具有 Permissions management（权限管理）访问级别分类的策略。

要查看分配给服务中每个操作的访问级别分类，请参阅 Amazon 服务的操作、资源和条件键。

服务摘要（操作列表）

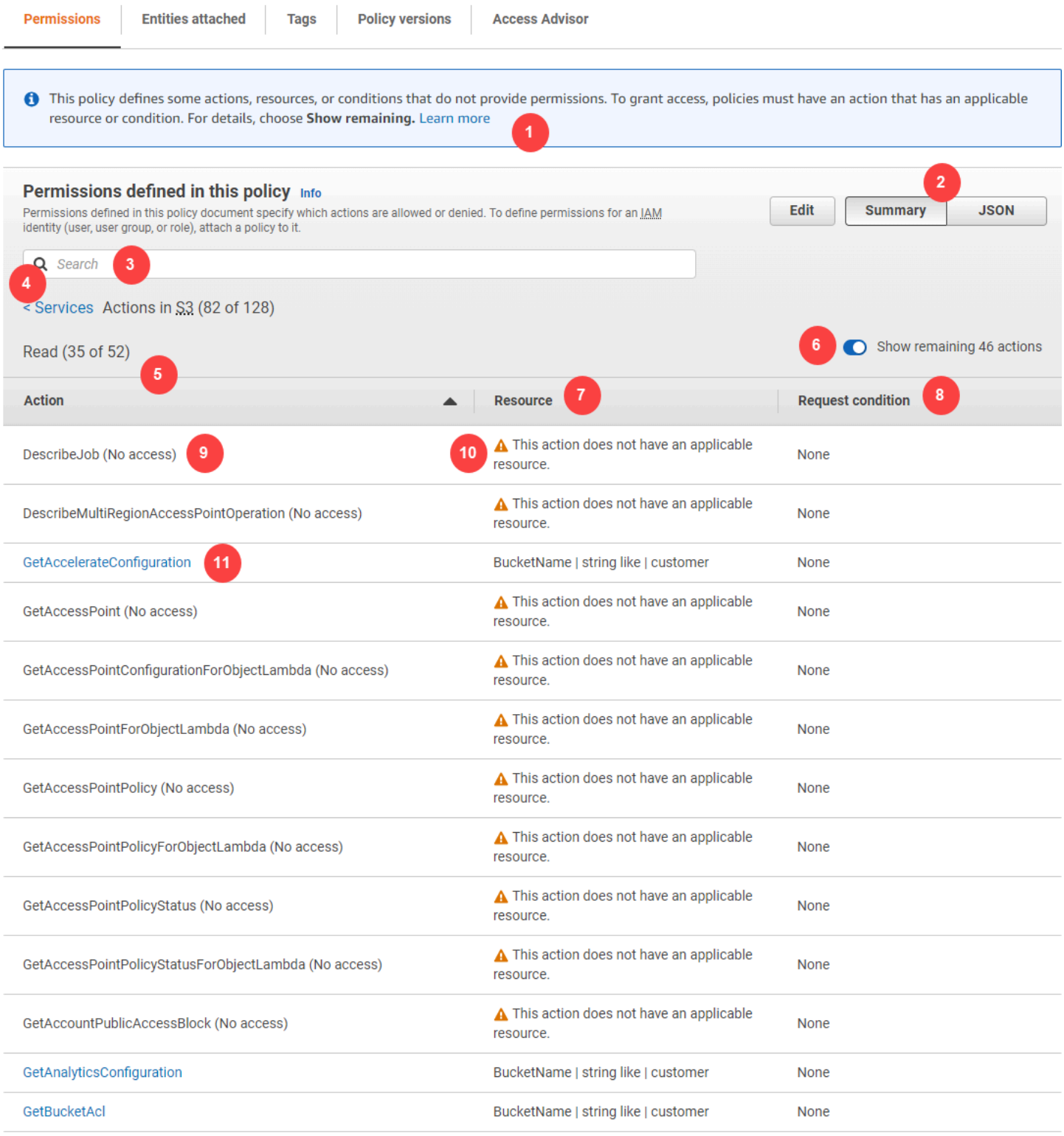
策略在三个表中概括：策略摘要、服务摘要和操作摘要。服务摘要表包括由所选服务的策略定义的操作列表和权限摘要。



对于策略摘要中列出的每个授予权限的服务，您都可以查看服务摘要。该表分组为 Uncategorized actions、Uncategorized resource types 和访问级别部分。如果策略包含 IAM 无法识别的操作，则该操作将包含在表的 Uncategorized actions（无法识别的操作）部分中。如果 IAM 能够识别该操作，则它将包含在表的某个访问级别（List（列出）、Read（读取）、Write（写入）和 Permissions management（权限管理））部分下。要查看分配给服务中每个操作的访问级别分类，请参阅 Amazon 服务的操作、资源和条件键。

了解服务摘要的元素

下面的示例是策略摘要许可的 Amazon S3 操作的服务摘要。此服务的操作按访问级别分组。例如，在服务可用的总计 52 个读取操作之中，定义了 35 个读取操作。



托管策略的服务摘要页面上包含以下信息：

1. 如果策略未向策略中服务定义的所有操作、资源和条件授予权限，则将在页面顶部显示警告或错误横幅。服务摘要中包含有关该问题的详细信息。要了解策略摘要如何帮助您了解策略授予的权限并进行相关问题排查，请参阅the section called “我的策略未授予预期权限”。
2. 选择 JSON 可查看策略的其他详细信息。您可以执行此操作以查看应用于操作的所有条件。(如果您要查看直接附加到用户的内联策略的服务摘要，则必须关闭服务摘要对话框并返回到策略摘要以访问 JSON 策略文档。)
3. 要查看特定操作的摘要，请在搜索框中键入关键字以缩短可用操作列表。
4. 在服务返回箭头旁将显示服务的名称（在此例中为 S3）。此服务的服务摘要包括在策略中定义的允许或拒绝的操作的列表。如果服务显示在权限选项卡的（显式拒绝）下，则服务摘要表中列出的操作将被明确拒绝。如果服务出现在权限选项卡的允许下，则服务摘要表中列出的操作将被允许。
5. 操作 – 此列将列出在策略中定义的操作，并提供每个操作的资源和条件。如果策略向操作授予或拒绝权限，则操作名称链接到操作摘要表。该表将操作分组为至少 1 个或最多 5 个部分，具体取决于策略允许或拒绝的访问权限级别。这些部分是列表、读取、写入、权限管理和标记。计数表示每个访问级别内提供权限的已识别操作的数量。总计是服务的已知操作数量。在此示例中，总共 52 个已知 Amazon S3 读取操作，有 35 个操作提供权限。要查看分配给服务中每个操作的访问级别分类，请参阅 Amazon 服务的操作、资源和条件键。
6. 显示剩余操作 – 切换此按钮可展开或隐藏该表，以包含已知但未向此服务提供权限的操作。切换按钮还将显示未提供权限的任何元素的警告。
7. Resource（资源）- 此列显示策略为服务定义的资源。IAM 不检查资源是否适用于每个操作。在此示例中，只允许对 `developer_bucket` Amazon S3 桶资源执行 Amazon S3 服务中的操作。根据服务提供给 IAM 的信息，您可能会看到一个 ARN（如 `arn:aws:s3:::developer_bucket/*`），或者您可能会看到定义的资源类型（如 `BucketName = developer_bucket`）。

   > **Note**
   >
   > 此列可以包括来自不同服务的资源。如果包含资源的策略语句不包括来自同一服务的操作和资源，则您的策略将包括不匹配的资源。在创建策略或在服务摘要中查看策略时，IAM 不会就不匹配的资源向您发出警告。IAM 也不会指示操作是否适用于资源，仅会指示服务是否匹配。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。为了更好地了解您的策略，请始终使用策略模拟器进行测试。

8. Request condition（请求条件）- 此列指出与资源关联的操作是否受条件约束。要了解有关这些条件的更多信息，请选择 JSON 以查看 JSON 策略文档。
9. (No access）（无权访问）- 此策略包含未提供权限的操作。
10. Resource warning（资源警告）对于资源不提供完全权限的操作，您将看到以下警告之一：

- 此操作不支持资源级权限。这需要为资源提供通配符 (*)。- 这意味着策略包含资源级权限，但必须包含 "Resource": ["*"] 来提供此操作的权限。
- 此操作没有适用的资源。- 这意味着策略中包含该操作，但没有支持资源。
- This action does not have an applicable resource and condition. (此操作没有适用的资源和条件。) - 这意味着策略中包含该操作，但没有支持资源和支持条件。在这种情况下，策略中还包含用于此服务的条件，但没有适用于此操作的条件。

11提供权限的操作包含指向操作摘要的链接。

查看服务摘要

对于策略摘要中列出的每个授予权限的服务，您都可以查看服务摘要。

从策略页面查看服务摘要

您可以在策略页面上查看管理型策略的服务摘要。

查看托管策略的服务摘要

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要查看的策略的名称。
4. 在策略的策略详细信息页面上，查看权限选项卡以查看策略摘要。
5. 在服务的策略摘要列表中，选择您想要查看的服务的名称。

查看附加到用户的策略的服务摘要

可以查看附加到 IAM 用户的任何策略的服务摘要。

查看附加到用户的策略的服务摘要

1. 登录 Amazon Web Services 管理控制台，单击 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在导航窗格中，选择用户。
3. 在用户列表中，选择要查看其策略的用户的名称。
4. 在用户的 Summary 页面上，查看 Permissions 选项卡，以查看直接附加到用户或从组附加到用户的策略列表。

5. 在用户的策略表中，选择要查看的策略的名称。

   如果您已打开用户页面，并选择查看附加到该用户的策略的服务摘要，则您会被重定向到策略页面。您只能在策略页面上查看服务摘要。

6. 选择摘要。在服务的策略摘要列表中，选择您想要查看的服务的名称。

> **Note**
>
> 如果您选择的策略是直接附加到用户的内联策略，则将显示服务摘要表。如果策略是附加到组的内联策略，则您将看到该组的 JSON 策略文档。如果策略是托管策略，则您将在 Policies 页面中看到该策略的服务摘要。

查看附加到角色的策略的服务摘要

可以查看附加到角色的任何策略的策略摘要。

查看附加到角色的策略的服务摘要

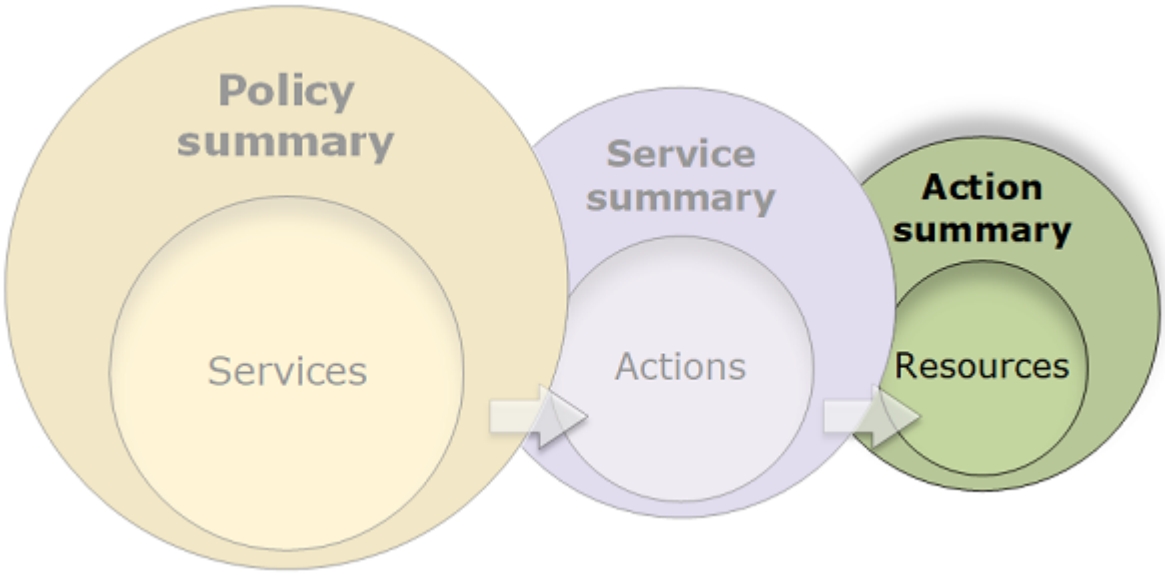
1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 从导航窗格中，选择 Roles。
3. 在角色列表中，选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上，查看 Permissions 选项卡，以查看附加到角色的策略列表。
5. 在角色的策略表中，选择要查看的策略的名称。

   如果您已打开角色页面，并选择查看附加到该用户的策略的服务摘要，则您会被重定向到策略页面。您只能在策略页面上查看服务摘要。

6. 在服务的策略摘要列表中，选择您想要查看的服务的名称。

操作摘要（资源列表）

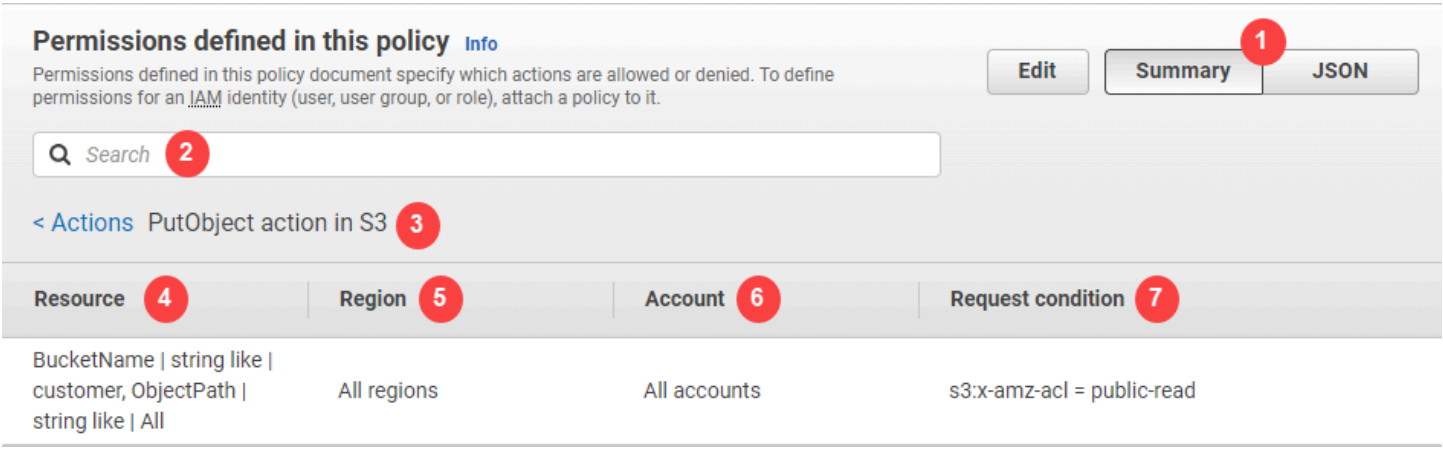
策略在三个表中概括：策略摘要、服务摘要和操作摘要。操作摘要 表包含应用至所选操作的资源和相关条件的列表。



要查看授予权限的每个操作的操作摘要，请在服务摘要中选择链接。操作摘要表包含资源的详细信息，其中包括资源的 Region 和 Account。您还可以查看适用于每个资源的条件。这将显示适用于某些资源而不是其他资源的条件。

了解操作摘要的元素

以下示例是 Amazon S3 服务摘要中 `PutObject`（写入）操作的操作摘要（请参阅 服务摘要（操作列表））。对于此操作，策略在单个资源上定义多个条件。



操作摘要页面包含以下信息：

1. 选择 JSON 可查看有关策略的其他详细信息，如查看适用于这些操作的多个条件。(如果您要查看直接附加到用户的内联策略的操作摘要，步骤则不同。这种情况下要访问 JSON 策略文档，则必须关闭服务摘要对话框并返回到策略摘要。)

2. 要查看特定资源的摘要，请在搜索框中键入关键字以缩短可用资源列表。
3. 操作返回箭头旁边显示的是 `action name action in service` 格式的服务和操作名称（在本例中为 S3 中的 PutObject 操作）。此服务的操作摘要包括在策略中定义的资源列表。
4. Resource（资源）- 此列将列出策略为所选服务定义的资源。在此示例中，PutObject 操作被允许在所有对象路径上执行，但仅在 `developer_bucket` Amazon S3 存储桶资源上使用。根据服务提供给 IAM 的信息，您可能会看到一个 ARN（如 `arn:aws:s3:::developer_bucket/*`），或者您可能会看到定义的资源类型（如 `BucketName = developer_bucket, ObjectPath = All`）。
5. Region（区域）- 该列显示在其中定义资源的区域。可以为所有区域或单个区域定义资源。它们不能位于多个特定的区域中。
   - 所有区域 – 与资源关联的操作适用于所有区域。在此示例中，该操作属于一项全球服务 Amazon S3。属于全球服务的操作适用于所有区域。
   - Region text - 与资源关联的操作适用于一个区域。例如，策略可以为资源指定 `us-east-2` 区域。
6. Account（账户）- 此列指示与资源相关联的服务或操作是否适用于特定账户。资源可以存在于所有账户中，也可以存在于单个账户中。它们不能存在于多个特定账户中。
   - All accounts（所有账户）- 与资源相关联的操作适用于所有账户。在此示例中，该操作属于一项全球服务 Amazon S3。属于全球服务的操作适用于所有账户。
   - 此账户 – 与资源相关联的操作仅适用于当前账户。
   - Account number - 与该资源相关联的操作适用于一个账户（不是您当前登录的账户）。例如，如果策略为资源指定 `123456789012` 的账户，则账号将显示在策略摘要中。
7. Request condition（请求条件）- 此列显示与资源关联的操作是否受条件约束。此示例包含 `s3:x-amz-acl = public-read` 条件。要了解有关这些条件的更多信息，请选择 JSON 以查看 JSON 策略文档。

查看操作摘要

可以查看策略摘要中列出的授予权限的每个操作的操作摘要。

从策略页面查看操作摘要

可以查看托管式策略的操作摘要。

查看托管策略的操作摘要

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要查看的策略的名称。
4. 在策略的策略详细信息页面上，查看权限选项卡以查看策略摘要。
5. 在服务的策略摘要列表中，选择您想要查看的服务的名称。
6. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

查看附加到用户的策略的操作摘要

可以查看附加到用户的任何策略的操作摘要。

查看附加到用户的策略的操作摘要

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 从导航窗格中选择 Users。
3. 在用户列表中，选择要查看其策略的用户的名称。
4. 在用户的 Summary 页面上，查看 Permissions 选项卡，以查看直接附加到用户或从组附加到用户的策略列表。
5. 在用户的策略表中，选择要查看的策略的名称。

   如果您已打开用户页面，并选择查看附加到该用户的策略的服务摘要，则您会被重定向到策略页面。您只能在策略页面上查看服务摘要。
6. 在服务的策略摘要列表中，选择您想要查看的服务的名称。

   > **Note**
   >
   > 如果您选择的策略是直接附加到用户的内联策略，则将显示服务摘要表。如果策略是附加到组的内联策略，则您将看到该组的 JSON 策略文档。如果策略是托管策略，则您将在 Policies 页面中看到该策略的服务摘要。
7. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

查看附加到角色的策略的操作摘要

可以查看附加到角色的任何策略的操作摘要。

查看附加到角色的策略的操作摘要

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择角色。
3. 在角色列表中，选择要查看其策略的角色的名称。
4. 在角色的 Summary 页面上，查看 Permissions 选项卡，以查看附加到角色的策略列表。
5. 在角色的策略表中，选择要查看的策略的名称。

   如果您已打开角色页面，并选择查看附加到该用户的策略的服务摘要，则您会被重定向到策略页面。您只能在策略页面上查看服务摘要。
6. 在服务的策略摘要列表中，选择您想要查看的服务的名称。
7. 在操作的服务摘要列表中，选择您想要查看的操作的名称。

策略摘要示例

以下示例包括 JSON 策略及其关联的策略摘要、服务摘要和操作摘要，可帮助您了解通过策略授予的权限。

策略 1：DenyCustomerBucket

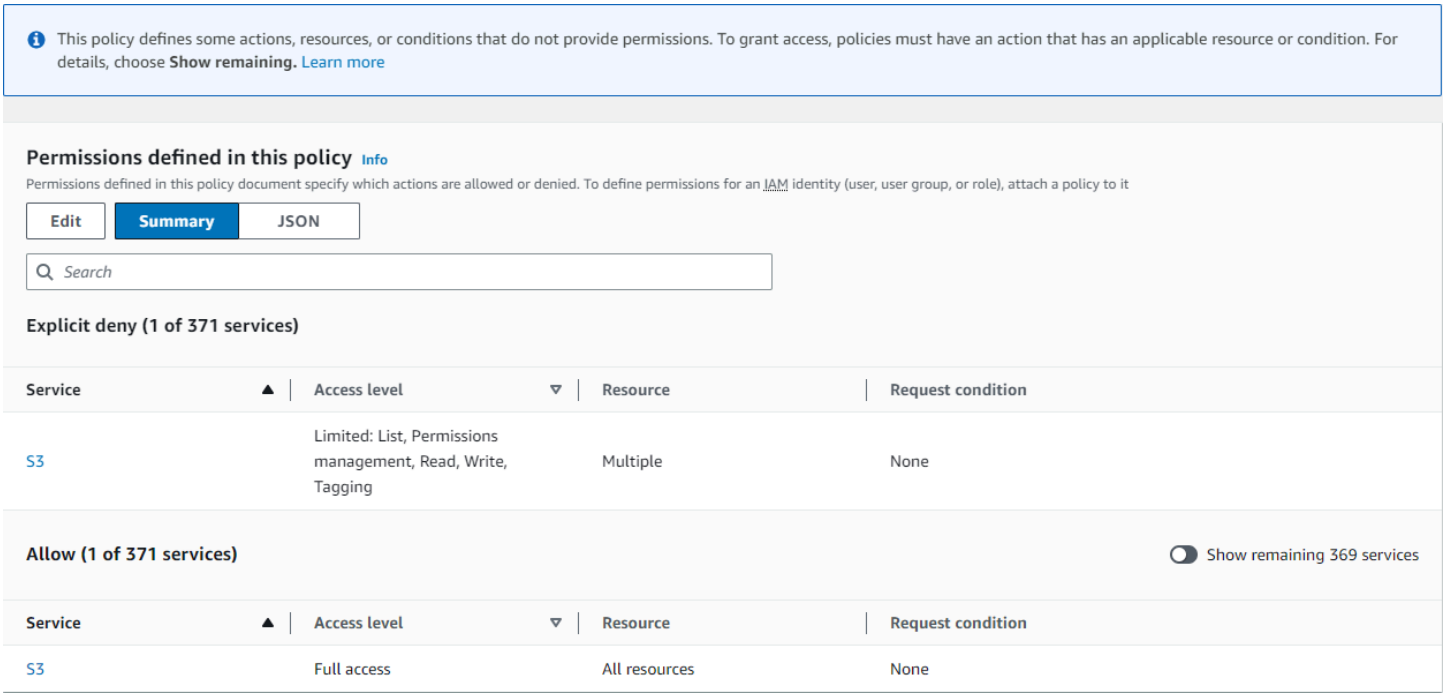
此策略展示对同一项服务的允许和拒绝。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "FullAccess",
            "Effect": "Allow",
            "Action": ["s3:*"],
            "Resource": ["*"]
```

```
        },
        {
            "Sid": "DenyCustomerBucket",
            "Action": ["s3:*"],
            "Effect": "Deny",
            "Resource": ["arn:aws:s3:::customer", "arn:aws:s3:::customer/*" ]
        }
    ]
}
```

DenyCustomerBucket 策略摘要：



DenyCustomerBucket S3 (Explicit deny) 服务摘要：

< Services Actions in S3 (82 of 130) Show remaining 48 actions

Read (35 of 53)

Action	Resource	Request condition
GetAccelerateConfiguration	BucketName| string like |customer	None
GetAnalyticsConfiguration	BucketName| string like |customer	None
GetBucketAcl	BucketName| string like |customer	None
GetBucketCORS	BucketName| string like |customer	None
GetBucketLocation	BucketName| string like |customer	None
GetBucketLogging	BucketName| string like |customer	None
GetBucketNotification	BucketName| string like |customer	None
GetBucketObjectLockConfiguration	BucketName| string like |customer	None
GetBucketOwnershipControls	BucketName| string like |customer	None
GetBucketPolicy	BucketName| string like |customer	None
GetBucketPolicyStatus	BucketName| string like |customer	None
GetBucketPublicAccessBlock	BucketName| string like |customer	None
GetBucketRequestPayment	BucketName| string like |customer	None
GetBucketTagging	BucketName| string like |customer	None
GetBucketVersioning	BucketName| string like |customer	None
GetBucketWebsite	BucketName| string like |customer	None

GetObject (Read) 操作摘要：

< Actions GetObject action in S3

Resource	Region	Account	Request condition
BucketName| string like |customer, ObjectPath| string like |All	-	All accounts	None

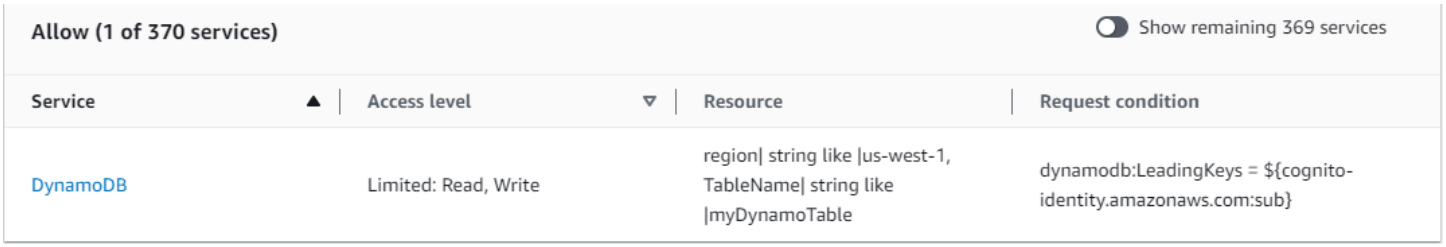
策略2：DynamoDbRowCognitoID

该策略基于用户的 Amazon Cognito ID 提供对 Amazon DynamoDB 的行级别访问权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "dynamodb:DeleteItem",
                "dynamodb:GetItem",
                "dynamodb:PutItem",
                "dynamodb:UpdateItem"
            ],
            "Resource": [
                "arn:aws:dynamodb:us-west-1:123456789012:table/myDynamoTable"
            ],
            "Condition": {
                "ForAllValues:StringEquals": {
                    "dynamodb:LeadingKeys": [
                        "${cognito-identity.amazonaws.com:sub}"
                    ]
                }
            }
        }
    ]
}
```

DynamoDbRowCognitoID 策略摘要：



DynamoDbRowCognitoID DynamoDB (允许) 服务摘要：

< Services Actions in DynamoDB (4 of 65) Show remaining 61 actions

Read (1 of 26)

Action	Resource	Request condition
GetItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

Write (3 of 33)

Action	Resource	Request condition
DeleteItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
PutItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}
UpdateItem	region| string like |us-west-1, TableName| string like |myDynamoTable	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

GetItem (List) 操作摘要：

< Actions GetItem action in DynamoDB

Resource	Region	Account	Request condition
region| string like |us-west-1, TableName| string like |myDynamoTable	us-west-1	123456789012	dynamodb:LeadingKeys = ${cognito-identity.amazonaws.com:sub}

策略 3：MultipleResourceCondition

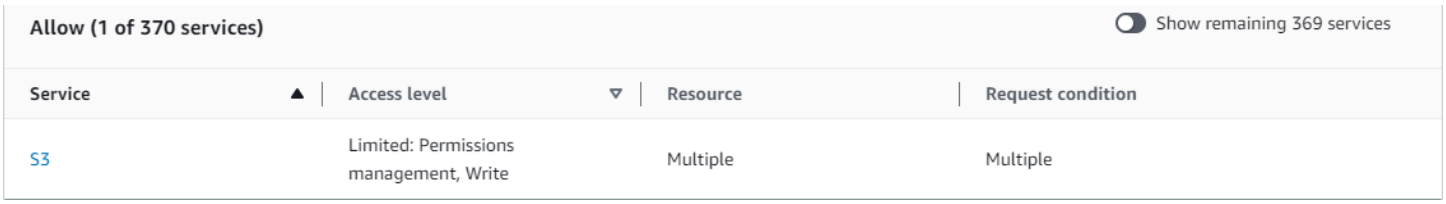
此策略包括多个资源和条件。

JSON

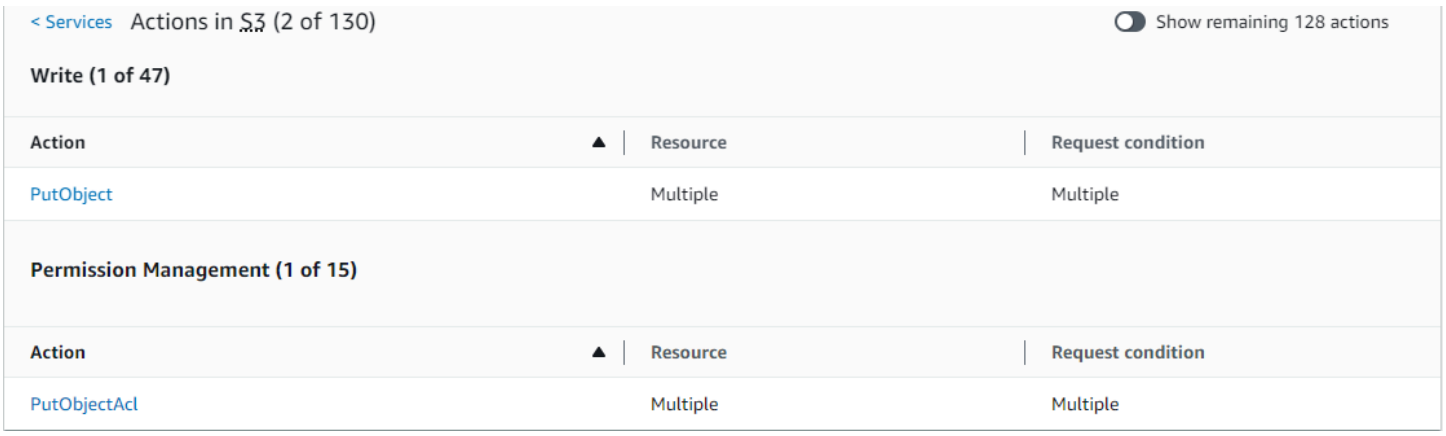
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Apple_bucket/*"],
            "Condition": {"StringEquals": {"s3:x-amz-acl": ["public-read"]}}
```

```
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": ["arn:aws:s3:::Orange_bucket/*"],
            "Condition": {"StringEquals": {
                "s3:x-amz-acl": ["custom"],
                "s3:x-amz-grant-full-control": ["1234"]
            }}
        }
    ]
}
```

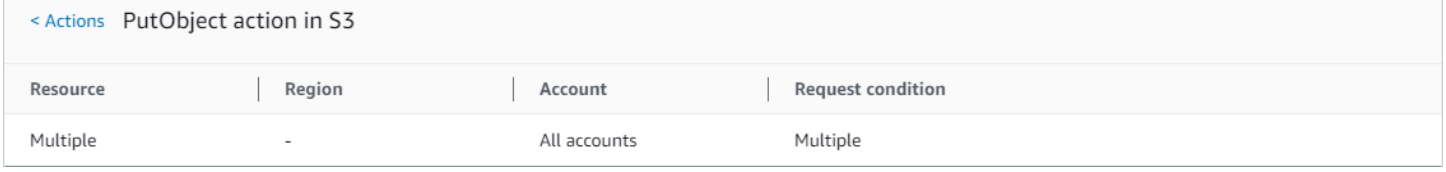
MultipleResourceCondition 策略摘要：



MultipleResourceCondition S3 (允许) 服务摘要：



PutObject (Write) 操作摘要：



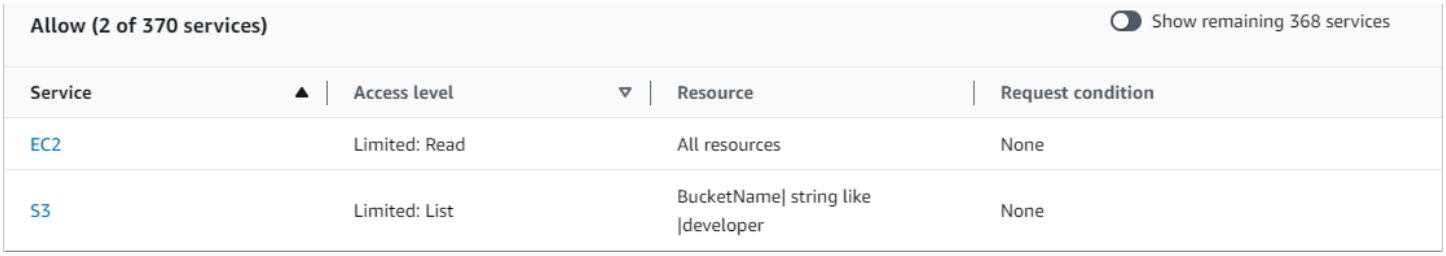
策略 4：EC2_troubleshoot

以下策略允许用户获取正在运行的 Amazon EC2 实例的截图，这可以帮助排查 EC2 故障。该策略还允许查看有关 Amazon S3 开发人员存储桶中的项目的信息。

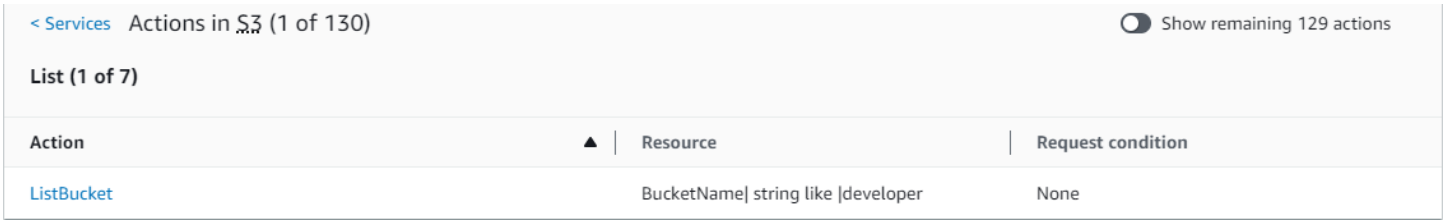
JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "ec2:GetConsoleScreenshot"
            ],
            "Resource": [
                "*"
            ]
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::developer"
            ]
        }
    ]
}
```

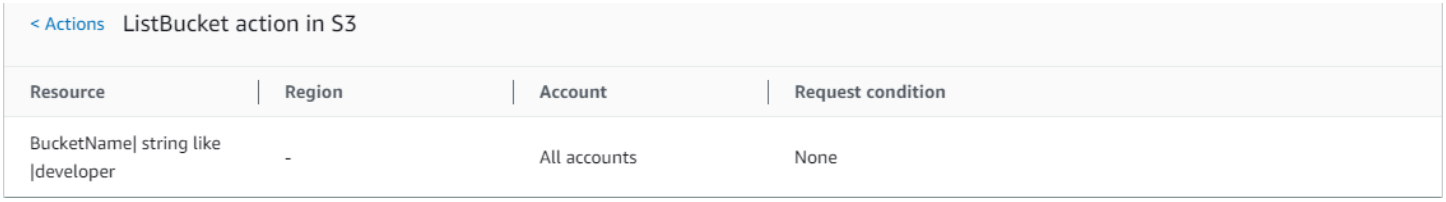
EC2_Troubleshoot 策略摘要：



EC2_Troubleshoot S3 (允许）服务摘要：



ListBucket (List) 操作摘要：



策略 5：CodeBuild_CodeCommit_CodeDeploy

此策略提供对特定 CodeBuild、CodeCommit 和 CodeDeploy 资源的访问。由于这些资源特定于每个服务，因此它们只与匹配的服务一起出现。如果您包含的资源与 `Action` 元素中的任何服务均不匹配，则该资源将出现在所有操作摘要中。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1487980617000",
            "Effect": "Allow",
            "Action": [
                "codebuild:*",
                "codecommit:*",
                "codedeploy:*"
            ],
            "Resource": [
                "arn:aws:codebuild:us-east-2:123456789012:project/my-demo-
project",
                "arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo",
                "arn:aws:codedeploy:us-
east-2:123456789012:application:WordPress_App",
                "arn:aws:codedeploy:us-east-2:123456789012:instance/AssetTag*"
            ]
        }
```

```
    ]
}
```

CodeBuild_CodeCommit_CodeDeploy 策略摘要：

Service	Access level	Resource	Request condition
Allow (3 of 103 services) Show remaining 100			
CodeBuild	**Limited**: List, Read, Write	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
CodeCommit	**Full**: Read, Write **Limited**: List	arn:aws:codecommit:us-east-2:123456789012:MyDemoRepo	None
CodeDeploy	**Limited**: List, Read, Write	Multiple	None

CodeBuild_CodeCommit_CodeDeploy CodeBuild (Allow) 服务摘要：

Action (9 of 15) Show remaining 6	Resource	Request condition
List (1 of 3 actions)		
ListBuildsForProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
Read (2 of 5 actions)		
BatchGetBuilds	codebuild:us-east-2:123456789012:project/my-demo-project	None
BatchGetProjects	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
Write (6 of 7 actions)		
BatchDeleteBuilds	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
CreateProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
DeleteProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
StartBuild	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
StopBuild	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None
UpdateProject	arn:aws:codebuild:us-east-2:123456789012:project/my-demo-project	None

CodeBuild_CodeCommit_CodeDeploy StartBuild (Write) 操作摘要：

Resource	Region	Account	Request condition
codebuild:us-east-2:123456789012:project/my-demo-project	us-east-2	123456789012	None

访问 IAM 资源所需的权限

Resources（资源）是服务中的对象。IAM 资源包括组、用户、角色和策略。如果您使用 Amazon Web Services 账户根用户 凭证进行登录，则在管理 IAM 凭证或 IAM 资源方面没有任何限制。不过，必须显式为 IAM 用户授予权限以管理凭证或 IAM 资源。您可以将基于身份的策略附加到用户以执行此操作。

> **Note**
>
> 在整个 Amazon 文档中，在提及 IAM policy 而未提到任何特定类别时，我们指的是基于身份的客户管理型策略。有关策略类别的详细信息，请参阅the section called “策略和权限”。

用于管理 IAM 身份的权限

管理 IAM 组、用户、角色和凭证所需的权限通常与该任务的 API 操作相对应。例如，要创建 IAM 用户，您必须具有执行相应 API 命令的 `iam:CreateUser` 权限：`CreateUser`。要允许 IAM 用户创建其他 IAM 用户，您可以将类似下面的 IAM policy 附加到该用户：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:CreateUser",
    "Resource": "*"
  }
}
```

在策略中，`Resource` 元素的值取决于操作以及操作能够影响的资源。在上述示例中，策略允许用户创建任何用户 (* 是与所有字符串匹配的通配符)。相反，仅允许用户更改自己的访问密钥的策略

（API 操作 CreateAccessKey 和 UpdateAccessKey）通常具有一个 Resource 元素。在此情况下，ARN 包含一个解析为当前用户名的变量 (${aws:username})，如以下示例中所示：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ListUsersForConsole",
            "Effect": "Allow",
            "Action": "iam:ListUsers",
            "Resource": "arn:aws:iam::*:*"
        },
        {
            "Sid": "ViewAndUpdateAccessKeys",
            "Effect": "Allow",
            "Action": [
                "iam:UpdateAccessKey",
                "iam:CreateAccessKey",
                "iam:ListAccessKeys"
            ],
            "Resource": "arn:aws:iam::*:user/${aws:username}"
        }
    ]
}
```

在上述示例中，${aws:username} 是解析为当前用户的用户名的变量。有关策略变量的更多信息，请参阅 IAM policy 元素：变量和标签。

一般来说，在操作名称中使用通配字符 (*) 可更轻松地为与特定任务相关的所有操作授予权限。例如，要允许用户执行任意 IAM 操作，您可以为操作使用 iam:*。要允许用户执行仅与访问密钥相关的操作，您可以在策略语句的 iam:*AccessKey* 元素中使用 Action。这可以授予用户执行 CreateAccessKey、DeleteAccessKey、GetAccessKeyLastUsed、ListAccessKeys 和 UpdateAccessKey 操作的权限。（如果未来在 IAM 中添加了名称包含“AccessKey”的操作，为 Action 元素使用 iam:*AccessKey* 还会授予用户执行该新操作的权限。）以下示例演示允许用户执行与自己的访问密钥有关的所有操作（将 *account-id* 替换为您的 Amazon Web Services 账户 ID）的策略：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::111122223333:user/${aws:username}"
    }
}
```

有些任务（例如删除组）涉及多个操作：必须先从组中删除用户，再分离或删除组的策略，然后才能真正删除组。如果您希望一个用户能够删除群组，必须确保授予该用户执行所有相关操作的权限。

在 Amazon Web Services 管理控制台中工作的权限

上述示例演示允许用户使用 Amazon CLI 或 Amazon 开发工具包执行操作的策略。

在用户使用控制台时，控制台向 IAM 发出请求以列出组、用户、角色和策略并获取与组、用户或角色关联的策略。控制台还会发出请求以获取 Amazon Web Services 账户 信息以及有关主体的信息。主体是在控制台中发出请求的用户。

通常，要执行某个操作，您必须仅在策略中包含匹配的操作。要创建用户，您需要具有调用 `CreateUser` 操作的权限。通常，在使用控制台执行操作时，您必须具有相应的权限才能在控制台中显示、列出、获取或以其他方式查看资源。您必须具有该权限，以便能够在控制台中导航以执行指定的操作。例如，如果用户 Jorge 要使用控制台更改自己的访问密钥，他可以访问 IAM 控制台并选择 Users（用户）。这种操作会导致控制台发出 `ListUsers` 请求。如果 Jorge 没有执行 `iam:ListUsers` 操作的权限，在控制台尝试列出用户时，将会被拒绝访问。因此，Jorge 无法获取自己的名称和访问密钥，即使他具有执行 `CreateAccessKey` 和 `UpdateAccessKey` 操作的权限。

如果要为用户授予权限以使用 Amazon Web Services 管理控制台管理组、用户、角色、策略和凭证，您需要包含控制台执行的操作的权限。有关可用于向用户授予这些权限的部分策略示例，请参阅管理 IAM 资源的策略示例。

向 Amazon 帐户授予权限

您可以直接向自己账户中的 IAM 用户授予对您的资源的访问权限。如果其他账户中的用户需要访问您的资源，您可以创建 IAM 角色，该角色是一个实体，其包含权限，但不与特定用户相关联。随后，其

他账户中的用户可以使用角色，并根据您分配给角色的权限来访问资源。有关更多信息，请参阅 在您拥有的其他 Amazon Web Services 账户 中 IAM 用户的访问权限。

> **Note**
>
> 有些服务支持基于资源的策略，如 基于身份的策略和基于资源的策略 中所述（例如 Amazon S3、Amazon SNS 和 Amazon SQS）。对于这些服务，使用角色的替代方法是将策略附加到要共享的资源（存储桶、主题或队列）。基于资源的策略可以指定具有访问资源权限的 Amazon 账户。

某种服务访问其他服务的权限

许多 Amazon 服务都要访问其他 Amazon 服务。例如，几个 Amazon 服务（包括 Amazon EMR、Elastic Load Balancing 和 Amazon EC2 Auto Scaling）负责管理 Amazon EC2 实例。其他 Amazon 服务利用 Amazon S3 存储桶、Amazon SNS 主题、Amazon SQS 队列等。

在这些情况下管理许可的方案根据服务的不同而各异。下面是一些如何针对不同服务处理许可的示例：

- 在 Amazon EC2 Auto Scaling 中，用户必须拥有使用 Auto Scaling 的权限，但无需明确获得管理 Amazon EC2 实例的权限。
- 在 Amazon Data Pipeline 中，IAM 角色决定了管道可以执行哪些操作；用户需要权限来担任角色。（有关详细信息，请参阅 Amazon Data Pipeline 开发人员指南中的使用 IAM 向管道授予权限。）

有关如何正确配置权限以便 Amazon 服务能够完成您打算执行的任务的详细信息，请参阅您调用服务的文档。要了解如何为服务创建角色，请参阅创建向 Amazon 服务委派权限的角色。

使用 IAM 角色配置服务以代表您工作

当要配置 Amazon 服务代表您工作时，您通常要为 IAM 角色提供 ARN，用于定义允许该服务执行哪些操作。Amazon 会检查并确保您有权将角色传递给该服务。有关更多信息，请参阅 向用户授予权限以将角色传递给 Amazon 服务。

所需的操作

操作是可以对资源执行的操作，例如，查看、创建、编辑和删除该资源。操作是由每个 Amazon 服务定义的。

要允许某个人执行操作，您必须在应用于进行调用的身份或受影响的资源的策略中包含所需的操作。通常，要提供执行某个操作所需的权限，您必须在策略中包括该操作。例如，要创建用户，您需要在策略中添加 CreateUser 操作。

在某些情况下，操作可能要求在策略中包含其他相关操作。例如，要为某个人提供在 Amazon Directory Service 中使用 `ds:CreateDirectory` 操作创建目录的权限，您必须在策略中包含以下操作：

- `ds:CreateDirectory`
- `ec2:DescribeSubnets`
- `ec2:DescribeVpcs`
- `ec2:CreateSecurityGroup`
- `ec2:CreateNetworkInterface`
- `ec2:DescribeNetworkInterfaces`
- `ec2:AuthorizeSecurityGroupIngress`
- `ec2:AuthorizeSecurityGroupEgress`

在使用可视化编辑器创建或编辑策略时，将显示警告和提示以帮助您选择您的策略所需的所有操作。

有关在 Amazon Directory Service 中创建目录所需的权限的更多信息，请参阅示例 2：允许用户创建目录。

管理 IAM 资源的策略示例

以下 IAM policy 示例允许用户执行与管理 IAM 用户、组和凭证相关的任务。这包括允许用户管理自己的密码、访问密钥和多重验证 (MFA) 设备的策略。

有关允许用户执行对其他 Amazon 服务（如 Amazon S3、Amazon EC2 和 DynamoDB）的任务的策略的示例，请参阅 IAM 基于身份的策略示例。

主题

- 允许用户列出账户的组、用户、策略等，以供报告之用
- 允许用户管理组的成员资格
- 允许用户管理 IAM 用户
- 允许用户设置账户密码策略
- 允许用户生成和检索 IAM 凭证报告

- 允许所有 IAM 操作（管理员访问）

允许用户列出账户的组、用户、策略等，以供报告之用

以下策略允许用户调用以字符串 Get 或 List 开头的任何 IAM 操作，并生成报告。要查看示例策略，请参阅IAM：允许对 IAM 控制台进行只读访问。

允许用户管理组的成员资格

以下策略允许用户更新名为 MarketingGroup 的组的成员资格。要查看示例策略，请参阅IAM：允许以编程方式和在控制台中管理组的成员资格。

允许用户管理 IAM 用户

以下策略允许用户执行所有与管理 IAM 用户相关的任务，但是不允许对其他实体执行操作，如创建组或策略。允许的操作包括这些：

- 创建用户（CreateUser 操作）。
- 删除用户。此任务需要授予执行以下所有操作的权限：DeleteSigningCertificate、DeleteLoginProfile、RemoveUserFromGroup 和 DeleteUser。
- 列出账户和组中的用户（GetUser、ListUsers 和 ListGroupsForUser 操作）。
- 列出和删除用户的策略（ListUserPolicies、ListAttachedUserPolicies、DetachUserPolicy、DeleteUserPolicy 操作）
- 重命名或更改用户的路径（UpdateUser 操作）。Resource 元素必须包括涉及源路径和目标路径的 ARN。有关路径的更多信息，请参阅易记名称和路径。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowUsersToPerformUserActions",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:ListPolicies",
                "iam:GetPolicy",
                "iam:UpdateUser",
                "iam:AttachUserPolicy",
                "iam:ListEntitiesForPolicy",
                "iam:DeleteUserPolicy",
                "iam:DeleteUser",
                "iam:ListUserPolicies",
                "iam:CreateUser",
                "iam:RemoveUserFromGroup",
                "iam:AddUserToGroup",
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:PutUserPolicy",
                "iam:ListAttachedUserPolicies",
                "iam:ListUsers",
                "iam:GetUser",
                "iam:DetachUserPolicy"
            ],
            "Resource": "*"
        },
        {
            "Sid": "AllowUsersToSeeStatsOnIAMConsoleDashboard",
            "Effect": "Allow",
            "Action": [
                "iam:GetAccount*",
                "iam:ListAccount*"
            ],
            "Resource": "*"
        }
    ]
}
```

上述策略中包含的很多权限允许用户在 Amazon Web Services 管理控制台中执行任务。仅从 Amazon CLI、Amazon SDK 或 IAM HTTP 查询 API 执行与用户相关的任务的用户可能不需要特定权限。例如，如果用户已知道从用户分离的策略的 ARN，则不需要 `iam:ListAttachedUserPolicies` 权限。用户所需权限的确切列表取决于用户管理其他用户时必须执行的任务。

以下策略中的权限允许通过 Amazon Web Services 管理控制台访问用户任务：

- `iam:GetAccount*`
- `iam:ListAccount*`

允许用户设置账户密码策略

您可以授予某些用户获取和更新您的 Amazon Web Services 账户 的密码策略的权限。要查看示例策略，请参阅IAM：允许以编程方式和在控制台中设置账户密码要求。

允许用户生成和检索 IAM 凭证报告

您可以授予用户生成和下载报告的权限，该报告列出了您 Amazon Web Services 账户 中的所有用户。该报告还列出了各种用户凭证的状态，包括密码、访问密钥、MFA 设备和签名证书。有关凭证报告的更多信息，请参阅为您的 Amazon Web Services 账户 生成凭证报告。要查看示例策略，请参阅IAM：生成和检索 IAM 凭证报告。

允许所有 IAM 操作（管理员访问）

您可以授予某些用户在 IAM 中执行所有操作的管理员权限，包括管理密码、访问密钥、MFA 设备和用户凭证。以下示例策略授予这些权限。

> **Warning**
>
> 当您向用户授予对 IAM 的完全访问权时，对用户可以向自己或他人授予的权限没有限制。用户可以创建新的 IAM 实体（用户或角色）并授予这些实体对您 Amazon Web Services 账户 中所有资源的完全访问权限。您向用户授予对 IAM 的完全访问权限时，实际上是向用户授予对您 Amazon Web Services 账户 中所有资源的完全访问权限。其中包括删除所有资源的权限。您应该仅将这些权限授予信任的管理员，还应对这些管理员强制采用多重身份验证 (MFA)。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "iam:*",
    "Resource": "*"
  }
}
```

IAM 临时委派

概述

临时委派可以加快 Amazon 和 Amazon 合作伙伴提供的与您的 Amazon 账户集成产品的载入速度并简化管理。您无需手动配置多项 Amazon 服务，而是可以委派临时的有限权限，允许产品提供商通过自动部署工作流程在几分钟内代表您完成设置任务。您可以通过批准要求和权限边界保持管理控制，而产品提供商的权限将在经过批准的持续时间后自动过期，无需手动清理。如果产品需要持续访问权限才能进行持续操作，则提供商可以使用临时委派来创建一个 IAM 角色，其权限边界定义了该角色的最大权限。所有产品提供商的活动都通过 Amazon CloudTrail 进行跟踪，以进行合规和安全监控。

> **Note**
>
> 临时委派请求只能由已完成功能载入过程的 Amazon 产品和符合资格的 Amazon 合作伙伴创建。客户可审核并批准这些请求，但无法直接创建请求。如果您是想要将 IAM 临时委派集成到产品中的 Amazon 合作伙伴，请参阅合作伙伴集成指南以获取载入和集成说明。

临时委派的工作原理

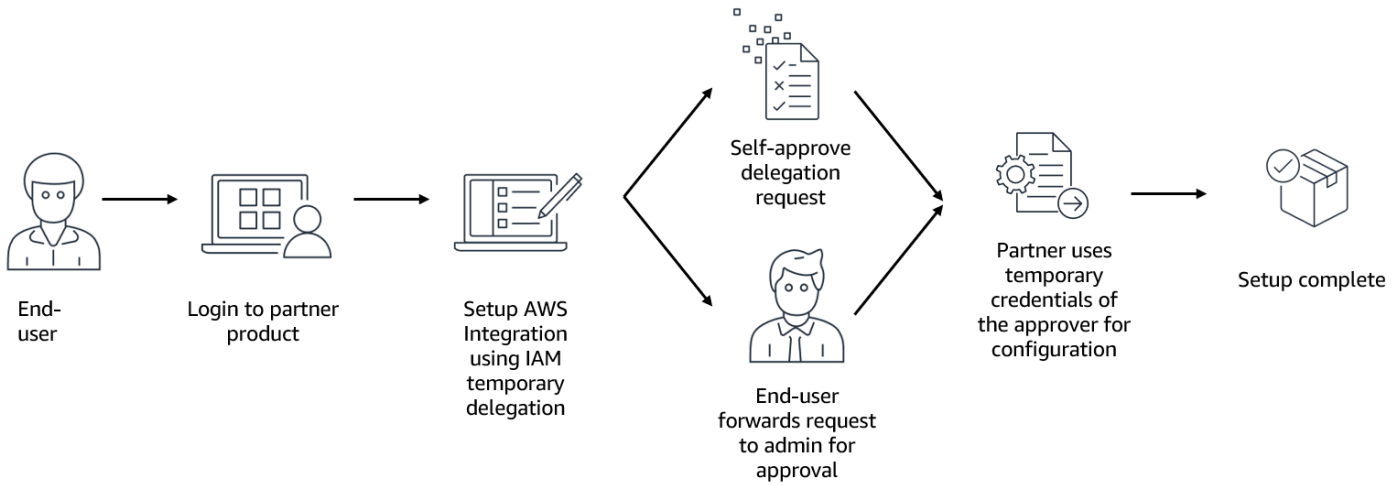
临时委派可让 Amazon 和 Amazon 合作伙伴请求对您的账户进行临时、有限的访问。在您批准后，他们可以使用委派权限代表您采取行动。委派请求定义了产品提供商在您的 Amazon 账户中部署或配置资源所需的 Amazon 服务和操作的特定权限。这些权限仅在有限的时间内可用，并在经过请求中指定的持续时间后自动过期。

> **Note**
>
> 委派访问的最长持续时间为 12 小时。但是，根用户只能批准持续时间不超过 4 小时的委派请求。如果请求指定的时间超过 4 小时，则必须使用非根用户身份来批准该请求。有关详细信息，请参阅权限模拟测试版功能。

对于正在进行的任务（例如从 Amazon S3 存储桶读取），委派请求可以包括创建一个 IAM 角色，允许在临时访问权限到期后继续访问资源和操作。产品提供商必须为通过临时委派创建的任何 IAM 角色附加权限边界。权限边界限制角色的最大权限，但不能自行授予权限。在批准请求之前，您可以查看作为请求一部分的权限边界。有关详细信息，请参阅权限边界。

具体流程如下：

1. 您可登录 Amazon 或 Amazon 合作伙伴产品以将其与您的 Amazon 环境集成。
2. 产品提供商代表您发起委派请求，并将您重定向到 Amazon 管理控制台。
3. 您可以查看所请求的权限，并决定是批准、拒绝还是将该请求转发给您的管理员。
4. 您或您的管理员批准请求后，产品提供商就可以获得批准者的临时凭证来执行所需任务。
5. 产品提供商访问权限将在经过指定的时间段后自动过期。但是，通过临时委派请求创建的任何 IAM 角色在该期限过后仍然存在，从而允许产品提供商继续访问资源和操作以执行持续的管理任务。



> **Note**
>
> 只有当您拥有临时委派请求中所包含服务和操作的权限时，您才能将权限委派给产品提供商。如果您无权访问请求的服务和操作，则在您批准请求后，产品提供商不会获得这些权限。

如果权限检查表明其很可能成功，则可以批准临时委派请求并继续该工作流程。

如果权限检查表明您可能没有足够的权限，请将请求转发给您的管理员进行批准。我们建议使用您偏好的方式（例如电子邮件或工单）将此请求通知您的管理员。

管理员批准请求后，接下来会发生什么取决于产品提供商的配置：

- 如果产品提供商请求立即访问，他们将自动获得临时权限，访问持续时间随即开始计时。
- 如果产品提供商请求所有者（初始接收者）发布，则您必须在访问持续时间开始之前返回该请求，以显式共享临时账户访问权限。当产品提供商需要您输入其他信息（例如资源选择或配置详细信息）才能完成所需任务时，其通常会使用此选项。

发起临时委派请求

您只能从受支持的 Amazon 或 Amazon 合作伙伴产品发起临时委派请求。在支持临时委派的工作流程中，系统将提示您授予产品提供商临时的有限权限，以在您的账户中配置所需的 Amazon 资源。这种自动化方法无需您手动配置这些资源，从而提供了更加简化的体验。

授予访问权限之前，您可以查看委派请求的详细信息，例如产品提供商所需的特定 IAM 角色、策略和 Amazon 服务。如果您有足够的权限，则可以自行批准请求，也可以将请求转发给您的账户管理员进行批准。所有产品提供商的访问权限均有时间限制，可以根据需要进行监控和撤销。

发起临时委派请求

1. 前往需要与您的 Amazon 账户集成的、Amazon 或 Amazon 合作伙伴提供的受支持产品的控制台。
2. 选择使用 IAM 临时委派进行部署。请注意，选项名称可能因支持的产品而有所不同。有关详细信息，请参阅产品提供商的文档。

> **Note**
>
> 如果您尚未登录 Amazon 管理控制台，会打开一个指向 Amazon 登录页面的新窗口。我们建议您在从产品控制台发起临时委派请求之前登录您的 Amazon 账户。有关如何根据用户类型和要访问的 Amazon 资源进行登录的更多信息，请参阅 Amazon 登录用户指南。

3. 查看请求详情以确认产品提供商的产品名称和 Amazon 账户。您还可以查看产品提供商代表您执行操作时将使用的 Amazon 身份。
4. 查看访问权限详细信息，了解批准此请求后将临时委派的权限。

   - 权限摘要部分提供由 AI 生成的高级概述，可帮助您了解可以访问哪些类别的 Amazon 服务以及可以在每项服务中执行哪些类型的操作。
   - 选择查看 JSON 以查看产品提供商需要在您的 Amazon 账户中部署的特定权限，包括访问范围和资源限制。
   - 如果产品提供商在临时委派请求中创建 IAM 角色，则必须为该角色附加权限边界。这些 IAM 角色拥有在请求的访问持续时间到期后继续允许访问资源和操作的权限。选择查看详细信息以查看权限边界，该边界定义角色可以拥有的最大权限。产品提供商将在创建过程中对角色应用其他策略，以定义其实际权限。这些策略可能看起来比权限边界更狭窄或更广泛，具体取决于产品提供商如何定义这些策略。但是，权限边界可以保证，无论角色附加哪些策略，该角色的有效权限绝不会超过您在请求批准期间看到的权限。有关更多信息，请参阅权限边界。

5. 查看权限模拟结果。权限模拟功能会根据请求中包含的权限自动评估您身份的权限。根据此分析，将显示一条建议，指示是使用您的当前身份批准请求，还是将请求转发给管理员。有关详细信息，请参阅权限模拟测试版功能。
6. 在对话框中，选择您希望如何继续。
   - 当您的身份具有足够的权限允许产品提供商代表您执行载入程序时，请选择允许访问。选择此选项后，产品提供商的访问持续时间将在您提供访问权限后开始。
   - 如果您的身份没有足够的权限允许产品提供商代表您执行载入程序，请选择请求批准。然后，选择创建审批请求。选择此选项后，将创建一个临时委派请求链接，您可以与账户管理员共享该链接。您的管理员可以访问 Amazon 管理控制台或使用访问链接审查临时委派请求，以批准请求并与请求者共享临时访问权限。

> **Note**
>
> 授予产品提供商访问权限需要两个操作：接受委派请求（`AcceptDelegationRequest`）和发放交换令牌（`SendDelegatedToken`）。当您批准请求后，Amazon 管理控制台会自动执行这两个步骤。如果您使用 Amazon CLI 或 API，则必须分别执行这两个步骤。

权限模拟功能（测试版）

收到临时委派请求时，您可以自行批准该请求，也可以将其转发给您的账户管理员进行批准。只有当您拥有临时委派请求中所包含服务和操作的权限时，您才能将权限委派给产品提供商。如果您无权访问所请求的服务和操作，那么即使请求中包含这些权限，产品提供商也无法获得这些权限。

例如，临时委派请求要求能够创建 Amazon S3 存储桶、启动和停止 Amazon EC2 中的实例以及代入 IAM 角色。批准该请求的身份可以启动和停止 Amazon EC2 中的实例，并代入 IAM 角色，但没有创建 Amazon S3 存储桶的权限。该身份批准请求时，即使临时委派请求中包含这些权限，产品提供商也无法创建 Amazon S3 存储桶。

由于您只能委派自己已经拥有的权限，因此在批准之前评估自己是否拥有所请求的权限至关重要。权限模拟测试版功能可将您的权限与请求中包含的权限进行比较，从而帮助进行此评估。评测表明您是可以使用当前身份批准请求，还是需要将其转发给管理员。如果分析无法验证您是否拥有足够的权限，请将该请求转发给管理员进行审查。此评测基于模拟权限分析，可能与您的真实 Amazon 环境有所不同，因此在继续之前请仔细检查请求的权限。

后续步骤

发起临时委派请求后，您可以在请求的整个生命周期中对其进行管理和监控。以下程序可帮助您跟踪、批准和控制临时访问权限：

- 审查临时委派请求：监控您的访问请求的状态，并查看有关批准或拒绝临时委派请求的请求详细信息。
- 撤销临时委派访问权限：在活动临时委派会话自然过期之前立即终止这些会话。

审查临时委派请求

发起临时委派请求后，您可以在 IAM 控制台中监控、批准和拒绝请求。临时委派请求页面提供所有请求的集中视图，包括待批准、已完成或已拒绝的请求。作为管理员，您可以审查这些请求以授予产品提供商访问 Amazon 资源的权限，也可以根据组织的安全策略、业务要求或合规性标准拒绝这些请求。这种可见性可帮助您跟踪产品提供商访问权限的生命周期，并保持对临时权限的监督。

> **Note**
>
> 您必须拥有 iam:AcceptDelegationRequest 权限才能批准临时委派请求。

批准临时委派请求

1. 登录 Amazon 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择临时委派请求。
3. 主页面将显示临时委派请求列表，包含以下信息：
   - 请求 ID：请求的唯一标识符
   - 状态：当前状态（待处理、已批准、已拒绝、已共享、已过期）
   - 请求者：与请求关联的产品提供商
   - 发起者：向产品提供商发起请求的账户中的 IAM 主体
   - 请求创建时间：提交请求的时间
   - 请求过期时间：请求过期或即将过期的时间
4. （可选）使用筛选选项按状态查看请求：
   - 所有请求：查看所有请求，无论状态如何

- 待处理：查看等待管理员批准的请求
- 已批准：查看已批准的请求
- 已共享：查看已共享访问权限的请求
- 已拒绝：查看已拒绝的请求及拒绝原因

5. 要查看有关特定请求的详细信息或查看待批准的请求，请选择请求 ID。
6. 查看详细的请求信息：
   - 产品提供商信息
   - 请求原因和理由
   - 请求持续时间
   - 请求 Amazon 权限
7. 如果您是审查待处理请求的管理员，请选择以下选项之一：
   - 要批准请求，请选择批准。在批准对话框中，您可以查看权限模拟的结果。有关更多信息，请参阅权限模拟测试版功能。确认访问持续时间和您的 Amazon 身份后，选择批准以授予访问权限。如果产品提供商请求立即访问，他们将自动获得临时权限，访问持续时间随即开始计时。否则，请通知发起请求的人员向产品提供商发放访问权限。
   - 要拒绝请求，请选择拒绝。
     - 在拒绝对话框中，提供明确的拒绝原因，以帮助请求者了解拒绝其请求的原因。
     - 选择拒绝以拒绝访问。
8. 请求列表会自动刷新以显示最新的状态信息。您还可以手动刷新页面以查看状态更新。

撤销临时委派访问权限

尽管产品提供商访问会话设计为经过批准的持续时间后自动过期，但在某些情况下，您可能需要立即终止访问。当出现安全问题、产品提供商的工作提前完成或业务需求发生变化时，撤销活跃的产品提供商访问权限可提供一种紧急控制机制。请求发起者和管理员都可以撤销访问权限以维护安全性和运营控制。

撤销临时委派访问权限

1. 登录 Amazon 管理控制台，并通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧导航窗格中，选择临时委派请求。

3. 找到希望撤销的访问会话的请求 ID。
4. 选择操作，然后选择撤销访问权限。
5. 在对话框中，选择撤销访问权限以确认希望立即终止访问会话。

撤销访问权限后，产品提供商将无法再访问您的 Amazon 资源。出于审计目的，撤销记录在 Amazon CloudTrail 中。

> **Important**
>
> 撤销访问权限会立即终止产品提供商访问会话。任何使用该访问权限正在进行的工作或进程都将中断。确保撤销不会中断关键操作。

> **Note**
>
> 您无法撤销使用根用户批准的请求的访问权限。Amazon 建议您避免使用根用户批准委派请求。改用具有适当权限的 IAM 角色。

管理委派请求的权限

管理员可以授予 IAM 主体权限，使其能够管理来自产品提供商的委派请求。当您想要将批准权限委派给组织中的特定用户或团队，或者需要控制谁可以对委派请求执行特定操作时，此功能非常有用。

以下 IAM 权限可用于管理委派请求：

权限	说明
iam:AssociateDelegationRequest	将未分配的委派请求与您的 Amazon 账户关联
iam:GetDelegationRequest	查看委派请求的详细信息
iam:UpdateDelegationRequest	将委派请求转发给管理员以供批准

权限	说明
iam:AcceptDelegationRequest	批准委派请求
iam:SendDelegationToken	批准后将交换令牌发放给产品提供商
iam:RejectDelegationRequest	拒绝委派请求
iam:ListDelegationRequests	列出您账户的委派请求

> **Note**
>
> 默认情况下，发起委派请求的 IAM 主体会自动获得管理该特定请求的权限。他们可以将其与自己的账户关联、查看请求详细信息、拒绝请求、将其转发给管理员进行批准、在管理员批准后将交换令牌发放给产品提供商以及列出其拥有的委派请求。

通知

IAM 临时委派与 Amazon 用户通知服务集成，可帮助您随时了解委派请求状态变更。对于需要审查和批准委派请求的管理员来说，通知服务尤其有用。

使用 Amazon 用户通知服务，您可以将警报配置为通过多种渠道发送，包括电子邮件、Amazon Simple Notification Service（SNS）、Amazon Chatbot for Slack 或 Microsoft Teams 以及 Amazon 控制台移动应用程序。这可确保在正确的时间通知正确的人员，从而能够更快地响应待批准的请求或感知访问权限变化。您还可以根据组织的需求和安全要求，自定义触发通知的事件。

可用通知事件

您可订阅以接收以下 IAM 临时委派事件的通知：

- IAM 临时委派请求已创建
- IAM 临时委派请求已分配
- IAM 临时委派请求待批准

- IAM 临时委派请求已拒绝
- IAM 临时委派请求已接受
- IAM 临时委派请求已最终确定
- IAM 临时委派请求已过期

配置 通知

要配置 IAM 临时委派事件的通知，请执行以下操作：

1. 打开 Amazon 用户通知服务控制台
2. 创建或更新通知配置
3. 选择 Amazon IAM 作为服务
4. 选择希望接收相关通知的委派请求事件
5. 配置发送渠道（电子邮件、Amazon Chatbot 等）

有关配置 Amazon 用户通知服务（包括设置发送渠道和管理通知规则）的详细说明，请参阅 Amazon 用户通知服务文档。

CloudTrail

产品提供商使用临时委派访问权限执行的所有操作都会自动记录在 Amazon CloudTrail 中。这为 Amazon 账户中的产品提供商活动提供了全面的可见性和可审计性。您可以确定产品提供商采取了哪些操作、操作发生的时间以及执行这些操作的产品提供商账户。

为帮助您区分自己的 IAM 主体采取的操作和具有委派访问权限的产品提供商采取的操作，CloudTrail 事件在 `userIdentity` 元素下包含了一个名为 `invokedByDelegate` 的新字段。此字段包含产品提供商的 Amazon 账户 ID，可以轻松筛选和审计所有委派的操作。

CloudTrail 事件结构

以下示例显示产品提供商使用临时委派访问权限执行操作的 CloudTrail 事件：

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:Role-Session-Name",
```

```
        "arn": "arn:aws:sts::111122223333:assumed-role/Role-Name/Role-Session-Name",
        "accountId": "111122223333",
        "accessKeyId": "[REDACTED:AWS_ACCESS_KEY]",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2024-09-09T17:50:16Z",
                "mfaAuthenticated": "false"
            }
        },
        "invokedByDelegate": {
            "accountId": "444455556666"
        }
    },
    "eventTime": "2024-09-09T17:51:44Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "GetUserPolicy",
    "awsRegion": "us-east-1",
    "requestParameters": {
        "userName": "ExampleIAMUserName",
        "policyName": "ExamplePolicyName"
    },
    "eventType": "AwsApiCall",
    "recipientAccountId": "111122223333"
}
```

`invokedByDelegate` 字段包含使用委派访问权限执行操作的产品提供商的 Amazon 账户 ID。在此示例中，账户 444455556666（产品提供商）在账户 111122223333（客户账户）中执行了一项操作。

Amazon 合作伙伴的 IAM 临时委派

概述

IAM 临时委派使 Amazon 客户能够通过交互式指导工作流程将 Amazon 合作伙伴产品无缝载入和/或集成到其 Amazon 环境中。客户可以授予 Amazon 合作伙伴有限的临时访问权限，以配置所需的 Amazon 服务，从而减少载入摩擦并加快价值实现时间。

IAM 临时委派使合作伙伴能够：

- 通过自动资源预置简化客户载入
- 通过消除手动配置步骤来降低集成复杂性
- 通过透明、经客户批准的权限建立信任
- 使用权限边界实现具有长期访问模式的持续运行

工作原理

1. 合作伙伴创建委派请求：合作伙伴创建请求，指定其需要哪些权限以及持续多长时间
2. Amazon 管理控制台中的客户审查：客户可以准确查看合作伙伴请求的权限和原因
3. 客户批准：客户批准请求并发放交换令牌。令牌将发送给此指定 SNS 主题的合作伙伴。
4. 合作伙伴接收临时凭证：合作伙伴交换临时 Amazon 凭证的令牌
5. 合作伙伴配置资源：合作伙伴使用凭证在客户账户中设置所需的资源

合作伙伴资格认证

要获得临时委派集成资格，合作伙伴必须满足以下要求：

- ISV Accelerate 参与：您必须加入 ISV Accelerate（ISVA）计划。
- Amazon Marketplace 上架商品：您的产品必须已在 Amazon Marketplace 上架，并带有“已在 Amazon 上部署”徽章。

激活流程

完成以下步骤以将临时委派集成到您的产品中：

1. 第 1 步：查看要求

   查看此文档以了解资格认证要求并填写下面的合作伙伴问卷。

2. 第 2 步：提交您的载入请求

   发送电子邮件至 aws-iam-partner-onboarding@amazon.com 或联系您的 Amazon 代表。请随附填好的合作伙伴问卷，包括下面表格中的所有必填字段。

3. 第 3 步：Amazon 验证和审查

Amazon 会：

- 验证您是否符合资格认证标准
- 审查您的策略模板和权限边界
- 提供您所提交构件的反馈

4. 第 4 步：优化您的策略

   回复 Amazon 反馈，根据需要提交更新的策略模板或权限边界。

5. 步骤 5：完成注册

   获得批准后，Amazon 将：

   - 为您指定的账户启用 API 访问权限
   - 共享您的策略模板和权限边界的 ARN（如果适用）

   载入完成后，您将收到确认信息。然后，您可以从注册的账户访问临时委派 API、CreateDelegationRequest 和 GetDelegatedAccessToken，并开始将委派请求工作流程集成到产品中。

合作伙伴问卷

下表列出合作伙伴载入所需的信息：

信息	说明	必填
合作伙伴中心账户 ID	您在 Amazon 合作伙伴中心注册 Amazon 账户的账户 ID。	是
PartnerId	Amazon 合作伙伴中心提供的合作伙伴 ID。	否
Amazon Marketplace 产品 ID	Amazon 合作伙伴中心所提供产品的产品 ID。	是
Amazon accountIDs	您要用于调用临时委派 API 的 Amazon 账户 ID 列表。此列表应包含您的生产账户和非生产/测试账户。	是

信息	说明	必填
合作伙伴名称	客户查看您的临时委派请求时，此名称会在 Amazon 管理控制台中向其显示。	是
联系电子邮件	可用于与您联系集成相关事宜的一个或多个电子邮件地址。	是
请求者域	您的域（例如 www.example.com）	是
集成描述	简要描述您希望使用此功能解决的使用案例。您可以包含指向您的文档或其他公开材料的参考链接。	是
架构图	展示您的集成使用案例的架构图。	否
策略模板	您必须为此功能注册至少一个策略模板。策略模板定义您要在客户 Amazon 账户中请求的临时权限。有关更多信息，请参阅“策略模板”部分。	是
策略模板名称	要注册的策略模板的名称。	是
权限边界	如果您希望使用临时权限在客户的账户中创建 IAM 角色，则必须向 IAM 注册权限边界。权限边界将附加到您创建的 IAM 角色上，以限制该角色的最大权限。您可以使用选定的 Amazon 托管策略作为权限边界，也可以注册新的自定义权限边界（JSON）。有关更多信息，请参阅“权限边界”部分。	否
权限边界名称	权限边界的名称。格式为：arn:aws:iam::partner:policy/permission_boundary/<partner_domain>/<policy_name>_<date> 策略名称必须包含创建日期作为后缀。创建权限边界后，该名称将无法更新。如果您使用现有的 Amazon 托管策略，请改为提供托管策略 ARN。	否
权限边界描述	权限边界的描述。创建权限边界后，此描述将无法更新。	否

了解您的集成

完成载入流程后，您可以构建与 IAM 临时委派的集成。完整的集成通常涉及三个主要工作类别：

1. 用户体验和工作流程设计

在合作伙伴应用程序中构建前端体验，指导客户完成临时委派工作流程。合作伙伴应用程序应该：

- 提供清晰的载入或配置流程，客户可以在其中授予临时访问权限。明确标注此操作，例如“使用 IAM 临时委派进行部署”。
- 使用 CreateDelegationRequest API 返回的控制台链接将客户重定向到 Amazon 管理控制台以查看和批准委派请求
- 提供适当的消息，说明正在请求哪些权限以及请求的原因。客户可以在委派请求详细信息页面上看到此消息。
- 客户在 Amazon 中完成批准后，处理其返回您应用程序的流程。

临时授权请求的最佳实践

在为您的合作伙伴应用程序实施临时授权时，请遵循以下定义的最佳实践，进而帮助客户验证请求的真实性和准确性。

1。在请求消息中包含用户可识别的上下文

您应该在授权请求消息中包含用户特定的上下文。此信息可帮助用户识别其现有合作伙伴工作流程中的请求并区分合法请求。请求消息中的建议信息：

- 客户在您的服务中的账户标识符或用户名
- 工作区名称、订阅 ID 或组织标识符
- 正在访问的特定资源（集群名称、项目名称、环境）
- 为此次授权尝试生成的唯一事务或请求标识符

请求消息示例：

```
Request from Partner A workspace "production-analytics"
Account: user@example.com
Workspace ID: 1234ABCD
Cluster: ml-training-cluster-01
Request ID: 1111-2222-3333-4444
```

2. 可选：在启动授权时包括 Amazon 账户 ID

如果客户账户 ID 可用，请将其包含在授权请求中。此验证步骤在客户的意图和授权令牌之间创建了额外的绑定。

3. 设计用于安全验证的请求消息

整理您的请求消息，使客户能够在授予访问权限之前自信地验证合法性。用户体验要求：

- 在重定向到 Amazon 之前，在应用程序界面中突出显示请求消息
- 使用清晰的描述性语言，直接与客户当前的操作相关
- 避免使用可能应用于任何授权请求的通用消息
- 包括足够的详细信息，让客户验证请求是否符合其预期工作流程
- 显示将获得授权访问权限的 Amazon 账户 ID
- 明确说明您的应用程序将请求哪些权限

在将客户重定向到 Amazon 同意屏幕之前，请在应用程序中加入确认通知：

```
## Redirecting to IAM Temporary Delegation
You are about to grant [Your Service Name] temporary access to your AWS account.

Before clicking "Allow" on the AWS consent screen:
• Verify the request details match your current action
• Confirm the AWS account ID matches your intended account
• Ensure you initiated this request from [Your Service Name]
```

4. 会话绑定建议

当 Amazon IAM 处理核心授权流时，请在您的应用程序中实施以下做法以增强会话完整性：

- 为每次授权尝试生成唯一的一次性请求标识符
- 将授权请求与客户的活动应用程序会话相关联
- 根据您的授权流程实施适当的请求到期时间
- 验证授权回调是否与原始请求上下文匹配
- 记录所有授权请求的启动和完成情况，以便进行安全监控

通过遵循这些最佳实践，您可以帮助创建更安全的临时授权体验。

2. API 集成

使用 IAM 临时委派 API 发送和管理委派请求。注册 Amazon 账户后，您可访问以下 API：

- IAM CreateDelegationRequest：为客户的 Amazon 账户创建委派请求。此 API 会返回一个控制台链接，您可以将客户重定向到该链接以审查和批准请求。
- Amazon STS GetDelegatedAccessToken：在客户批准您的委派请求后检索临时 Amazon 凭证。使用这些凭证在客户的账户中执行操作。

您的集成应处理委派请求的整个生命周期，包括创建请求、监控其状态以及在获得批准后检索临时凭证。

3. 资源配置与编排

获得临时凭证后，编排必要的工作流程以配置客户 Amazon 账户中的资源。这可能包括：

- 直接调用 Amazon 服务 API 创建和配置资源
- 使用 Amazon CloudFormation 模板部署基础设施
- 创建用于持续访问的 IAM 角色（需要使用权限边界）

您的编排逻辑应具有幂等性，并且可以从容地处理故障，因为客户可能需要重试或修改其委派批准。

了解权限

作为功能载入流程的一部分，您需要向 IAM 注册策略，这些策略定义您要在客户 Amazon 账户中请求的权限。注册流程为客户提供了更加一致的体验，并有助于避免策略制定中的常见陷阱。

在注册期间，Amazon 根据一组验证来评估您的策略。这些验证旨在标准化策略格式和结构，并提供针对已知反模式的基本保护。验证还可以降低权限升级、意外跨账户访问以及广泛访问客户账户中高价值资源的风险。

权限类型

Amazon 将考虑两类权限：临时权限和长期权限。

临时权限

临时权限限制分配给任何临时委派访问会话的权限。临时权限如应用于委派会话的策略模板中所述。这些模板支持您在创建委派请求时提供的参数。这些参数值随后将绑定到会话。临时权限的工作方式与

Amazon STS 目前提供的会话策略相同：这些策略限制了底层用户的能力，但不授予任何其他访问权限。有关更多信息，请参阅 Amazon STS 文档中的会话策略。

长期权限

长期权限限制通过临时访问创建或管理的任何角色的权限。长期权限以 IAM 权限边界的形式实现。作为载入的一部分，您可以向 Amazon 提交一个或多个权限边界。获得批准后，Amazon 将与您共享一个策略 ARN，供您在策略中引用。

这些边界策略有两个值得注意的特征。首先，它们是不可变的。如果要更新权限，您可以注册新的权限边界。然后，您可以通过发送新的委派请求将新权限边界附加到客户的角色。其次，这些策略并非模板化。由于相同的边界策略是全局共享的，因此无法根据每个客户进行更改。

> **Important**
>
> 权限边界的最大大小限制为 6144 个字符。

> **Note**
>
> 如果您想更新权限边界或策略模板，请通过 aws-iam-partner-onboarding@amazon.com 与 IAM 联系。注册新的权限边界后，您可以向客户发送委派请求，以更新 IAM 角色并附加新注册的权限边界。有关更多详细信息，请参阅“示例”部分。

示例使用案例：数据处理工作负载

假设产品提供商在客户账户中运行数据处理工作负载。该提供商需要在初始载入期间设置基础设施，还需要持续访问权限才能运行工作负载。

临时权限（用于初始设置）：

- 创建 Amazon EC2 实例、VPC 和安全组
- 创建用于处理后数据的 Amazon S3 存储桶
- 创建用于持续运行的 IAM 角色
- 将权限边界附加到 IAM 角色

长期权限（用于持续运行、具有权限边界的 IAM 角色）：

- 启动和停止 Amazon EC2 实例以运行处理作业
- 从 Amazon S3 存储桶读取输入数据
- 将处理的结果写入 Amazon S3 存储桶

临时权限仅在载入期间使用一次，用于配置基础设施。在此过程中创建的 IAM 角色具有权限边界，其最大权限仅限于持续工作负载管理所需的操作。这样便可确保即使修改了角色的策略，也不会超过边界中定义的权限。

策略评估指南

Amazon 将根据一组指南评估您提交的策略。相同的评估指南适用于策略模板和权限边界，但会酌情注明细微的差异。

出于评估目的，我们将服务分为不同的组。最重要的区别在于管理访问权限、凭证和密钥安全敏感型服务。授予对这些服务访问权限的策略需要严格限定于正在进行的工作。安全敏感型服务包括以下服务：Amazon Identity and Access Management（IAM）、Amazon Key Management Service（KMS）、Amazon Resource Access Manager（RAM）、Amazon IAM Identity Center、Amazon Organizations 和 Amazon Secrets Manager。

第二个区别是可以跨账户边界访问数据的服务。这些服务的策略必须包括保护措施，以防止意外的跨账户访问。

通用验证

所有策略语句必须遵循以下指南：

- 所有语句都必须依次包含 Effect、Action（或 NotAction）、Resource 和 Condition 字段
- 单个语句中的所有操作必须按字母顺序列出
- 策略中包含的所有 ARN 都必须遵循相关服务公共文档中定义的语法
- NotAction 字段只能在 Deny 语句中使用
- Allow 语句中的 Action 必须包含服务代码。不允许使用通用通配符 ("*")

安全敏感型服务限制

以下限制适用于上述安全敏感型服务：

- Allow 语句中的 Action 必须比 [service] 更具体：*

- 临时访问策略模板的 Allow 语句中的 Action 不得包含通配符
- iam:PassRole 或 iam:CreateServiceLinkedRole 等敏感操作需要额外的范围限制，例如特定资源或条件检查。这些操作包括：
  - IAM 角色传递
  - IAM 角色修改操作
  - IAM 策略修改操作
  - Amazon KMS 写入或加密操作
  - Amazon RAM 写入或共享操作
  - 用于检索或修改密钥或者修改资源策略的 Amazon Secrets Manager 操作
- 可能使用通配符资源的其他操作，例如 iam:ListUsers 或 iam:GetPolicy
- 管理凭证的操作（例如 iam:CreateAccessKey）将被阻止

IAM 特定限制

对于 IAM：

- 只允许对 IAM 角色和策略执行有限的写入操作。您无法请求对用户、组和证书等其他 IAM 资源的权限。
- 策略附件或内联策略管理操作仅限于具有权限边界的角色。权限边界必须由合作伙伴提供，或者位于允许的 Amazon 托管策略列表中。如果 Amazon 托管策略未授予高权限或管理权限，则可能会允许这些策略。例如，针对特定工作职能的 Amazon 托管策略或 SecurityAudit 策略可能可接受。Amazon 将在载入过程中根据具体情况审查每个 Amazon 托管策略。
- 只允许对具有以下合作伙伴特定路径的策略进行策略管理：arn:aws:iam::@{AccountId}:policy/partner_domain.com/[feature]*
- 标签只能在资源创建期间应用，并且仅用于角色和策略
- iam:PassRole 检查必须匹配特定名称或路径前缀

Amazon STS 特定限制

对于 Amazon STS：

- sts:AssumeRole 必须限定为特定角色 ARN、角色 ARN 前缀，或仅限于一组账户或组织 ID /组织单位

IAM Identity Center 限制

对于 Amazon IAM Identity Center，将阻止以下操作：

- 所有涉及权限管理的操作（例如，sso:AttachCustomerManagedPolicyReferenceToPermissionSet）
- Amazon Identity Store 的用户、组和成员资格修改
- 标签管理

Amazon Organizations 限制

对于 Amazon Organizations，只允许读取操作。

其他服务特定验证

- 获取密钥或凭证的操作（例如 glue:GetConnection 或 redshift:GetClusterCredentials）必须包含匹配完整 ARN、ARN 前缀或标签的条件
- 对于 Amazon Redshift：redshift:GetClusterCredentials 只允许对特定数据库名称使用，而 redshift:GetClusterCredentialsWithIAM 只允许对特定工作组名称使用

> **Note**
>
> 管理账户中的 IAM 资源时，我们建议使用包含您名称的路径，例如 arn:aws:iam::111122223333:role/partner.com/rolename。这将有助于区分与您的集成关联的资源，让客户更轻松地进行发现、审计和分析。

跨账户访问要求

可能允许跨账户访问的语句必须至少包含以下各项之一：

- 指定资源账户或组织的条件（例如，与一个或多个预期值匹配的 aws:ResourceOrgId）
- 包含特定账户的资源字段（例如，arn:aws:sqs:*:111122223333:*）
- 包含非通配符账户和完整资源名称（例如，arn:aws:s3:::full-bucket-name）的资源字段

> **Note**
>
> 跨账户访问是一项敏感功能，需要明确的业务理由。Amazon 将在载入过程中仔细审查是否需要跨账户访问。

策略模板

策略模板是一种新的 IAM 构造，旨在定义合作伙伴在客户账户中请求的临时权限。与常规 IAM 策略一样，其使用包含 Effect、Action、Resource 和 Condition 元素的语句来定义权限。关键区别在于，策略模板包含的参数（例如 @{bucketName}）在您创建委派请求时会替换为实际值。

策略模板工作原理

作为载入过程的一部分，您可以向 Amazon 注册策略模板。Amazon 为每个模板分配一个唯一的 ARN，您在创建委派请求时会引用该 ARN。

创建委派请求时，您将指定：

- 策略模板 ARN
- 要替换到模板中的参数值

Amazon 将模板与您的参数值相结合，生成标准 IAM 策略。客户在批准您的委派请求时会审查这一最终呈现的策略，从而确切地了解将授予哪些权限。

> **Note**
>
> 最终呈现策略的最大大小限制为 2048 个字符。

以下是一个简单的示例，展示模板替换的工作原理。

策略模板：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
```

```
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::@{bucketName}/*"
        }
    ]
}
```

委派请求中提供的参数：

```
{
    "Name": "bucketName",
    "Values": ["customer-data-bucket"],
    "Type": "String"
}
```

最终呈现的策略（客户看到的内容）：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:PutObject"
            ],
            "Resource": "arn:aws:s3:::customer-data-bucket/*"
        }
    ]
}
```

模板语法

策略模板使用两项关键功能来提供灵活性：参数替换和条件语句。参数替换允许您在模板中定义占位符，这些占位符在创建委派请求时会替换为实际值。条件语句允许您根据参数值包含或排除整个策略语句。

参数替换和类型

使用 @{parameterName} 语法在策略模板中定义参数。创建委派请求时，您必须指定每个参数的类型。

字符串

直接替换到模板中的单个值。

模板：

```
"Resource": "arn:aws:s3:::@{bucketName}/*"
```

参数：

```
{
    "Name": "bucketName",
    "Values": ["my-bucket"],
    "Type": "String"
}
```

呈现的结果：

```
"Resource": "arn:aws:s3:::my-bucket/*"
```

StringList

生成多个资源条目的多个值。在资源 ARN 中使用 StringList 参数时，它会扩展为针对每个值创建单独的资源条目。

模板：

```
"Resource": "arn:aws:s3:::@{bucketNames}/*"
```

参数：

```
{
    "Name": "bucketNames",
    "Values": ["bucket-1", "bucket-2"],
    "Type": "StringList"
}
```

呈现的结果：

```
"Resource": [
```

```
    "arn:aws:s3:::bucket-1/*",
    "arn:aws:s3:::bucket-2/*"
]
```

交叉乘积行为

在同一个资源 ARN 中使用多个参数时，StringList 参数会创建所有组合的交叉乘积。

模板：

```
"Resource": "arn:aws:s3:::@{bucketNames}/@{prefix}/*"
```

参数：

```
[
    {
        "Name": "bucketNames",
        "Values": ["bucket-1", "bucket-2"],
        "Type": "StringList"
    },
    {
        "Name": "prefix",
        "Values": ["data"],
        "Type": "String"
    }
]
```

呈现的结果：

```
"Resource": [
    "arn:aws:s3:::bucket-1/data/*",
    "arn:aws:s3:::bucket-2/data/*"
]
```

条件语句

使用 @Enabled 指令可根据参数值有条件地包含或排除整个语句。

语法：

- @Enabled: "parameterName"：参数值为“True”时包含语句

- @Enabled: "!parameterName"：参数值不为“True”（否定）时包含语句

模板：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3:GetObject"],
            "Resource": "*"
        },
        {
            "@Enabled": "ENABLE_S3_WRITE",
            "Effect": "Allow",
            "Action": ["s3:PutObject"],
            "Resource": "arn:aws:s3:::@{bucketName}/*"
        }
    ]
}
```

参数（ENABLE_S3_WRITE 为“True”时）：

```
[
    {
        "Name": "bucketName",
        "Values": ["my-bucket"],
        "Type": "String"
    },
    {
        "Name": "ENABLE_S3_WRITE",
        "Values": ["True"],
        "Type": "String"
    }
]
```

呈现的结果：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
```

```
            "Effect": "Allow",
            "Action": ["s3:GetObject"],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": ["s3:PutObject"],
            "Resource": "arn:aws:s3:::my-bucket/*"
        }
    ]
}
```

参数（ENABLE_S3_WRITE 为“False”时）：

```
[
    {
        "Name": "bucketName",
        "Values": ["my-bucket"],
        "Type": "String"
    },
    {
        "Name": "ENABLE_S3_WRITE",
        "Values": ["False"],
        "Type": "String"
    }
]
```

呈现的结果：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": ["s3:GetObject"],
            "Resource": "*"
        }
    ]
}
```

ENABLE_S3_WRITE 设置为“True”时，将包含条件语句。设置为“False”时，将从呈现的策略中排除该语句。

其他示例

以下示例展示在临时委派中使用策略模板的常见模式。其重点介绍如何创建具有权限边界的 IAM 角色以实现长期访问，并展示将权限范围限定为特定资源的不同策略。这些示例说明如何使用 ARN 前缀、资源标记和权限边界更新等技术，在灵活性与安全性之间取得平衡。

示例 1：授予对特定资源的长期访问权限

以下权限边界作为“partner.com”的“SQSAccessorBoundary”提交：

```
{
    "Effect": "Allow",
    "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:*:*:*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
}
```

> **Note**
>
> 这包括一个同账户条件，以避免向其他账户中具有开放资源策略的队列授予访问权限。由于该边界在所有客户之间共享且无法进行模板化，因此不能直接引用客户的账户 ID。

由于这是此策略的第一个版本，其 ARN 为 arn:aws:iam::partner:policy/permissions-boundary/partner.com/SQSAccessorBoundary_2025_01_15

已提交以下策略模板用于获取临时访问权限：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
```

```
            "Action": [
                "sqs:ListQueues"
            ],
            "Resource": "arn:aws:sqs:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:PutRolePermissionsBoundary",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::@{AccountId}:role/partner.com/SQSAccessor",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::partner:policy/
permissions-boundary/partner.com/SQSAccessorBoundary_2025_01_15"
                }
            }
        }
    ]
}
```

示例 2：使用 ARN 前缀

权限边界可以指定资源 ARN 前缀来限制访问权限：

```
"Resource": "arn:aws:sqs:*:@{AccountId}:PartnerPrefix*"
```

这可将访问仅限于带有该前缀的资源，从而缩小可访问资源的范围。

示例 3：使用标签进行资源访问控制

您可以在临时委派的访问期间为资源添加标签，并依靠这些标签进行长期访问控制。

允许访问带标签资源的权限边界：

```
{
    "Effect": "Allow",
    "Action": [
        "sqs:DeleteMessage",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
```

```
    ],
    "Resource": "arn:aws:sqs:*:*:*",
    "Condition": {
        "Null": {
            "aws:ResourceTag/ManagedByPartnerDotCom": "false"
        },
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
}
```

在创建新队列时为其添加标签的策略模板：

```
{
    "Effect": "Allow",
    "Action": [
        "sqs:CreateQueue",
        "sqs:TagQueue"
    ],
    "Resource": "arn:aws:sqs:*:*:*",
    "Condition": {
        "Null": {
            "aws:RequestTag/ManagedByPartnerDotCom": "false"
        }
    }
}
```

为既有队列添加标签并创建角色的策略模板：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "sqs:TagQueue"
            ],
            "Resource": "arn:aws:sqs:*:@{AccountId}:@{QueueName}",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": "ManagedByPartnerDotCom"
                }
```

```
            }
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:CreateRole",
                "iam:PutRolePermissionsBoundary",
                "iam:PutRolePolicy"
            ],
            "Resource": "arn:aws:iam::@{AccountId}:role/partner.com/SQSAccessor",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::partner:policy/
permissions-boundary/partner.com/SQSAccessorBoundary_2025_01_15"
                }
            }
        }
    ]
}
```

这种方法允许客户显式确认可以长期访问哪些特定资源。

示例 4：更新权限边界

要更新权限边界，请注册带有新日期后缀的新版本，并请求替换该版本的权限。

具有额外权限的更新权限边界：

```
{
    "Effect": "Allow",
    "Action": [
        "sqs:DeleteMessage",
        "sqs:PurgeQueue",
        "sqs:ReceiveMessage",
        "sqs:SendMessage"
    ],
    "Resource": "arn:aws:sqs:*:*:*",
    "Condition": {
        "StringEquals": {
            "aws:ResourceAccount": "${aws:PrincipalAccount}"
        }
    }
}
```

作为第二个版本，此策略的 ARN 为：arn:aws:iam::partner:policy/permissions-boundary/partner.com/SQSAccessorBoundary_2025_01_20

更新现有角色权限边界的策略模板：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePermissionsBoundary"
            ],
            "Resource": "arn:aws:iam::@{AccountId}:role/partner.com/SQSAccessor",
            "Condition": {
                "StringEquals": {
                    "iam:PermissionsBoundary": "arn:aws:iam::partner:policy/
permissions-boundary/partner.com/SQSAccessorBoundary_2025_01_20"
                }
            }
        }
    ]
}
```

客户必须批准此委派请求，才能更新现有角色的权限边界。

构建您的集成

了解请求生命周期

在构建集成之前，了解委派请求从创建到完成的过程非常重要。

请求状态

委派请求会经历以下状态：

州	说明
未分配	请求已创建，但尚未与客户账户和 IAM 主体关联。该请求可能是在未指定目标账户的情况下创建的，也可能已指定目标账户 ID 但账户所有者尚未认领。

州	说明
已分配	与客户账户关联且正在等待审查的请求
待批准	客户已将请求转发给管理员进行批准
已接受	请求已获得客户批准，但交换令牌尚未发放
已最终确定	交换令牌已发放给产品提供商。委派期限（交换令牌有效期）从请求达到“已最终确定”状态时开始
已拒绝	请求被客户拒绝
已过期	由于不活动或超时，请求已过期

状态变换

正常流程（批准路径）

- 未分配 → 已分配：客户将请求与其账户关联
- “已分配 → 已接受”或“已分配 → 待批准”：客户直接批准请求或转发给管理员进行审查
- 待批准 → 已接受：管理员批准该请求
- 已接受 → 已最终确定：客户发放交换令牌

拒绝路径

- 已分配 → 已拒绝：客户拒绝该请求
- 待批准 → 已拒绝：管理员拒绝该请求
- 已接受 → 已拒绝：客户在发放令牌之前撤销批准

到期路径

如果在指定时间范围内未采取任何行动，则请求将自动过期：

- 未分配 → 已过期（1 天）
- 已分配 → 已过期（7 天）
- 待批准 → 已过期（7 天）

- 已接受 → 已过期（7 天）
- 已拒绝 → 已过期（7 天）
- 已最终确定 → 已过期（7 天）

终端状态

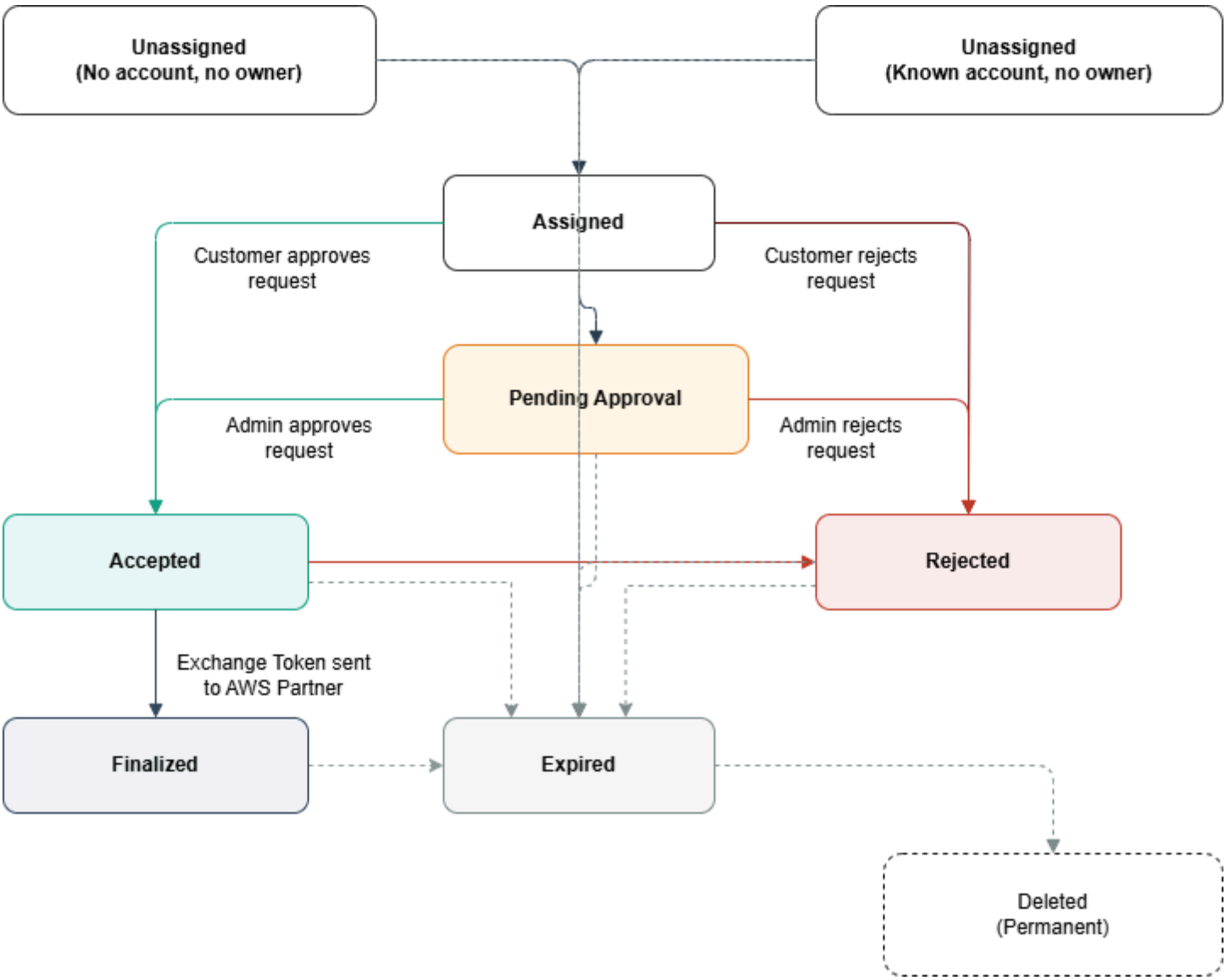
以下状态是终止状态（不再有进一步的转换）：

- 已最终确定：交换令牌已发送
- 已拒绝：请求被拒绝
- 已过期：请求超时或委派期限结束

保留期过后，过期的请求最终会从系统中删除。

管理应用程序中的委派请求状态

作为合作伙伴，您必须在系统中跟踪委派请求状态，并将其呈现给客户。当您收到有关状态变更的 SNS 通知时，请将这些更新存储在后端，并在面向客户的 UI 中显示。请特别注意“待批准”状态：当客户将请求转发给管理员进行审查时，Amazon 会向您发送待批准通知。在等待管理员操作期间，请求可以保持此状态长达 7 天。在此期间，在您的应用程序中向客户展示其请求正在等待管理员的批准。考虑提供指向 Amazon 管理控制台的深层链接，客户可以在其中查看请求状态或与其管理员跟进沟通。正确处理后端的状态机并在每个阶段向客户呈现正确的状态信息对于获得良好的集成体验非常重要。



配置 通知

IAM 使用 Amazon Simple Notification Service（SNS）向您传达委派请求状态的更改。创建委派请求时，必须提供注册 Amazon 账户中的 SNS 主题 ARN。IAM 将向该主题发布重要事件的消息，包括客户批准或拒绝请求时以及交换令牌准备就绪时。

> **Note**
>
> SNS 主题不能位于可选择加入的 Amazon 区域。您的 SNS 主题必须位于默认启用的 Amazon 区域中。有关选择加入区域的列表，请参阅 Amazon 账户管理指南中的“管理 Amazon 区域”。

SNS 主题配置

要接收委派请求通知，您必须将 SNS 主题配置为授予向其发布消息的 IAM 权限。将以下策略语句添加到 SNS 主题策略：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowIAMServiceToPublish",
            "Effect": "Allow",
            "Principal": {
                "Service": "iam.amazonaws.com"
            },
            "Action": "SNS:Publish",
            "Resource": "arn:aws:sns:REGION:ACCOUNT-ID:TOPIC-NAME"
        }
    ]
}
```

> **Important**
>
> SNS 主题必须位于您的其中一个注册 Amazon 账户中。IAM 不接受来自其他账户的 SNS 主题。如果主题策略配置不正确，您将不会收到状态更改通知或交换令牌。

通知类型

IAM 发送两种类型的通知：

StateChange 通知

当委派请求转换到新状态（已分配、待批准、已接受、已最终确定、已拒绝、已过期）时发送。

ExchangeToken 通知

当客户发放委派令牌时发送（状态为“已最终确定”）。此通知包含您获取凭证所需的交换令牌。

通知系统

您将收到以下委派请求状态的通知：

州	通知类型	说明
已分配	StateChange	请求已与客户账户关联
待批准	StateChange	客户已将请求转发给管理员进行批准
已接受	StateChange	客户已批准请求，但尚未发放令牌
已最终确定	StateChange	客户已发放交换令牌
已最终确定	ExchangeToken	此通知包含交换令牌
REJECTED	StateChange	客户已拒绝该请求
EXPIRED	StateChange	请求在完成前已过期

通知消息格式

IAM 发布标准 SNS 通知。委派请求信息以 JSON 字符串的形式包含在“消息”字段中。

常用字段（所有通知）

字段	Type	说明
类型	字符串	“StateChange”或“ExchangeToken”
RequestId	字符串	IAM 委派请求 ID
Requestor WorkflowId	字符串	您在创建请求时提供的工作流程 ID
州	字符串	请求的当前状态
OwnerAccountId	字符串	客户的 Amazon 账户 ID
UpdatedAt	字符串	状态更改时的时间戳（ISO 8601 格式）

其他字段（仅限 ExchangeToken 通知）

字段	Type	说明
ExchangeToken	字符串	使用 Amazon STS GetDelegatedAccessToken API 交换凭证的令牌
ExpiresAt	字符串	委派访问权限过期时间（ISO 8601 格式）

示例通知

StateChange 通知

```
{
  "Type": "Notification",
  "MessageId": "61ee8ad4-6eec-56b5-8f3d-eba57556aa13",
  "TopicArn": "arn:aws:sns:us-east-1:123456789012:partner-notifications",
  "Message": "{\"RequestorWorkflowId\":\"workflow-12345\",\"Type\":\"StateChange\",
\"RequestId\":\"dr-abc123\",\"State\":\"ACCEPTED\",\"OwnerAccountId\":\"111122223333\",
\"UpdatedAt\":\"2025-01-15T10:30:00.123Z\"}",
  "Timestamp": "2025-01-15T10:30:00.456Z",
  "SignatureVersion": "1",
  "Signature": "...",
  "SigningCertURL": "...",
  "UnsubscribeURL": "..."
}
```

ExchangeToken 通知

```
{
  "Type": "Notification",
  "MessageId": "e44e5435-c72c-5333-aba3-354406782f5b",
  "TopicArn": "arn:aws:sns:us-east-1:123456789012:partner-notifications",
  "Message": "{\"RequestId\":\"dr-abc123\",\"RequestorWorkflowId\":\"workflow-12345\",
\"State\":\"FINALIZED\",\"OwnerAccountId\":\"111122223333\",\"ExchangeToken\":
\"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...\",\"ExpiresAt\":\"2025-01-15T18:30:00.123Z\",
\"UpdatedAt\":\"2025-01-15T10:30:00.456Z\",\"Type\":\"ExchangeToken\"}",
  "Timestamp": "2025-01-15T10:30:00.789Z",
  "SignatureVersion": "1",
  "Signature": "...",
  "SigningCertURL": "...",
  "UnsubscribeURL": "..."
```

```
}
```

交换令牌

当客户接受并最终确定委派请求时，IAM 会发布交换令牌或交易令牌。产品提供商使用此交换或交易令牌调用 Amazon STS GetDelegatedAccessToken API，以获取具有客户批准权限的临时 Amazon 凭证。交换令牌本身不授予对 Amazon 资源的访问权限；必须通过 Amazon STS 将其交换为实际凭证。

交换令牌只能由创建委派请求的产品提供商账户兑换。请求账户嵌入在令牌中，确保只有授权的产品提供商才能获得访问客户账户的凭证。

访问持续时间

委派期限从客户发放交换令牌时开始，而不是从产品提供商兑换令牌时开始。客户发放令牌后：

- 产品提供商通过 SNS 通知接收令牌
- 其可以立即用该令牌交换凭证
- 凭证到期时间：发放时间 + 批准的持续时间
- 如果需要，产品提供商可以在令牌到期前多次交换该令牌以获取新的凭证

多次兑换

产品提供商可以在有效期内多次交换令牌以获得新的凭证。但是，根据您发放令牌的时间，从同一个交换令牌获得的所有凭证会同时过期。

示例：如果您批准了 2 小时的委派请求并在上午 10:00 发放令牌：

令牌发放时间	令牌交换时间	凭证到期时间	可用时间
10:00 am	10:00 am	12:00 pm	2 小时
10:00 am	10:20 am	12:00 pm	1 小时 40 分钟
10:00 am	11:40 am	12:00 pm	20 分钟
10:00 am	12:10 pm	失效（令牌已过期）	0 分钟

如表中所示，在有效期内较晚的时间交换令牌会导致产品提供商的可用时间缩短。

使用 Amazon 开发工具包的 IAM 代码示例

以下代码示例显示如何将 IAM 与 Amazon 软件开发工具包（SDK）一起使用。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用 Amazon 开发工具包的 IAM 代码示例
  - 使用 Amazon SDK 的 IAM 基础知识示例
    - 开始使用 IAM
    - 了解将 IAM 与 Amazon SDK 结合使用的基础知识
    - 使用 Amazon 开发工具包的 IAM 操作
      - 将 AddClientIdToOpenIdConnectProvider 与 CLI 配合使用
      - 将 AddRoleToInstanceProfile 与 CLI 配合使用
      - 将 AddUserToGroup 与 CLI 配合使用
      - 将 AttachGroupPolicy 与 CLI 配合使用
      - 将 AttachRolePolicy 与 Amazon SDK 或 CLI 配合使用
      - 将 AttachUserPolicy 与 Amazon SDK 或 CLI 配合使用
      - 将 ChangePassword 与 CLI 配合使用
      - 将 CreateAccessKey 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateAccountAlias 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateGroup 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateInstanceProfile 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateLoginProfile 与 CLI 配合使用
      - 将 CreateOpenIdConnectProvider 与 CLI 配合使用
      - 将 CreatePolicy 与 Amazon SDK 或 CLI 配合使用
      - 将 CreatePolicyVersion 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateRole 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateSAMLProvider 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
      - 将 CreateUser 与 Amazon SDK 或 CLI 配合使用

- 将 CreateVirtualMfaDevice 与 CLI 配合使用
- 将 DeactivateMfaDevice 与 CLI 配合使用
- 将 DeleteAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountPasswordPolicy 与 CLI 配合使用
- 将 DeleteGroup 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteGroupPolicy 与 CLI 配合使用
- 将 DeleteInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteLoginProfile 与 CLI 配合使用
- 将 DeleteOpenIdConnectProvider 与 CLI 配合使用
- 将 DeletePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeletePolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRolePermissionsBoundary 与 CLI 配合使用
- 将 DeleteRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSigningCertificate 与 CLI 配合使用
- 将 DeleteUser 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteUserPermissionsBoundary 与 CLI 配合使用
- 将 DeleteUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteVirtualMfaDevice 与 CLI 配合使用
- 将 DetachGroupPolicy 与 CLI 配合使用
- 将 DetachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DetachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 EnableMfaDevice 与 CLI 配合使用
- 将 GenerateCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GenerateServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetAccessKeyLastUsed 与 Amazon SDK 或 CLI 配合使用

- 将 GetAccountAuthorizationDetails 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountPasswordPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountSummary 与 Amazon SDK 或 CLI 配合使用
- 将 GetContextKeysForCustomPolicy 与 CLI 配合使用
- 将 GetContextKeysForPrincipalPolicy 与 CLI 配合使用
- 将 GetCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GetGroup 与 CLI 配合使用
- 将 GetGroupPolicy 与 CLI 配合使用
- 将 GetInstanceProfile 与 CLI 配合使用
- 将 GetLoginProfile 与 CLI 配合使用
- 将 GetOpenIdConnectProvider 与 CLI 配合使用
- 将 GetPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 GetRole 与 Amazon SDK 或 CLI 配合使用
- 将 GetRolePolicy 与 CLI 配合使用
- 将 GetSamlProvider 与 CLI 配合使用
- 将 GetServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 GetServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetServiceLastAccessedDetailsWithEntities 与 CLI 配合使用
- 将 GetServiceLinkedRoleDeletionStatus 与 Amazon SDK 或 CLI 配合使用
- 将 GetUser 与 Amazon SDK 或 CLI 配合使用
- 将 GetUserPolicy 与 CLI 配合使用
- 将 ListAccessKeys 与 Amazon SDK 或 CLI 配合使用
- 将 ListAccountAliases 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedGroupPolicies 与 CLI 配合使用
- 将 ListAttachedRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedUserPolicies 与 CLI 配合使用
- 将 ListEntitiesForPolicy 与 CLI 配合使用
- 将 ListGroupPolicies 与 CLI 配合使用

- 将 ListGroups 与 Amazon SDK 或 CLI 配合使用

- 将 ListGroupsForUser 与 CLI 配合使用
- 将 ListInstanceProfiles 与 CLI 配合使用
- 将 ListInstanceProfilesForRole 与 CLI 配合使用
- 将 ListMfaDevices 与 CLI 配合使用
- 将 ListOpenIdConnectProviders 与 CLI 配合使用
- 将 ListPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListPolicyVersions 与 Amazon SDK 或 CLI 配合使用
- 将 ListRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListRoleTags 与 CLI 配合使用
- 将 ListRoles 与 Amazon SDK 或 CLI 配合使用
- 将 ListSAMLProviders 与 Amazon SDK 或 CLI 配合使用
- 将 ListServerCertificates 与 Amazon SDK 或 CLI 配合使用
- 将 ListSigningCertificates 与 CLI 配合使用
- 将 ListUserPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListUserTags 与 CLI 配合使用
- 将 ListUsers 与 Amazon SDK 或 CLI 配合使用
- 将 ListVirtualMfaDevices 与 CLI 配合使用
- 将 PutGroupPolicy 与 CLI 配合使用
- 将 PutRolePermissionsBoundary 与 CLI 配合使用
- 将 PutRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 PutUserPermissionsBoundary 与 CLI 配合使用
- 将 PutUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 RemoveClientIdFromOpenIdConnectProvider 与 CLI 配合使用
- 将 RemoveRoleFromInstanceProfile 与 CLI 配合使用
- 将 RemoveUserFromGroup 与 CLI 配合使用
- 将 ResyncMfaDevice 与 CLI 配合使用
- 将 SetDefaultPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 TagRole 与 CLI 配合使用
- 将 TagUser 与 CLI 配合使用
- 将 UntagRole 与 CLI 配合使用

- 将 UntagUser 与 CLI 配合使用
- 将 UpdateAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 UpdateAccountPasswordPolicy 与 CLI 配合使用
- 将 UpdateAssumeRolePolicy 与 CLI 配合使用
- 将 UpdateGroup 与 CLI 配合使用
- 将 UpdateLoginProfile 与 CLI 配合使用
- 将 UpdateOpenIdConnectProviderThumbprint 与 CLI 配合使用
- 将 UpdateRole 与 CLI 配合使用
- 将 UpdateRoleDescription 与 CLI 配合使用
- 将 UpdateSamlProvider 与 CLI 配合使用
- 将 UpdateServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UpdateSigningCertificate 与 CLI 配合使用
- 将 UpdateUser 与 Amazon SDK 或 CLI 配合使用
- 将 UploadServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UploadSigningCertificate 与 CLI 配合使用

- 使用 Amazon 开发工具包的 IAM 场景
  - 使用 Amazon 开发工具包构建和管理弹性服务
  - 配置 Amazon ECS Service Connect
  - 使用 Lambda 代理集成创建 REST API
  - 创建 Fargate 启动类型的 Amazon ECS Linux 任务
  - 使用 Amazon 开发工具包创建只读和读写 IAM 用户
  - 将函数名称作为变量创建 CloudWatch 控制面板
  - 创建 EC2 启动类型的 Amazon ECS 服务
  - 创建 Amazon Managed Grafana 工作区
  - 创建第一个 Lambda 函数
  - 通过 CLI 开始使用 Redshift Serverless
  - IoT Device Defender 入门
  - 开始使用 Amazon EKS
  - Amazon MSK 入门
  - Amazon Redshift 预置集群入门

- Amazon SageMaker Feature Store 入门
- Config 入门
- Step Functions 入门
- 使用 Amazon 开发工具包管理 IAM 访问密钥
- 使用 Amazon 开发工具包管理 IAM policy
- 使用 Amazon 开发工具包管理 IAM 角色
- 使用 Amazon 开发工具包管理您的 IAM 账户
- 将硬编码的密钥移动到 Secrets Manager
- 允许 Amazon Compute Optimizer 自动化功能应用建议的操作
- 在整个组织中启用自动化功能的策略
- 启用账户自动化功能的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 授予计算优化自动化功能的服务相关角色权限的策略
- 使用 Amazon SDK 回滚 IAM policy 版本
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 使用 Amazon Command Line Interface v2 为 DynamoDB 设置基于属性的访问控制
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数
- 使用 Amazon Command Line Interface v2 处理 DynamoDB Streams 和生存时间
- 通过 Amazon SDK 使用 IAM Policy Builder API

- 使用 Amazon 开发工具包的 Amazon STS 代码示例
  - 使用 Amazon SDK 的 Amazon STS 基础知识示例
    - 使用 Amazon 开发工具包的 Amazon STS 操作
      - 将 AssumeRole 与 Amazon SDK 或 CLI 配合使用
      - 将 AssumeRoleWithWebIdentity 与 CLI 配合使用
      - 将 DecodeAuthorizationMessage 与 CLI 配合使用
      - 将 GetFederationToken 与 CLI 配合使用

        - 将 GetSessionToken 与 Amazon SDK 或 CLI 配合使用
- 使用 Amazon 开发工具包的 Amazon STS 场景
    - 使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色
    - 配置 Amazon ECS Service Connect
    - 使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL
    - 使用 Lambda 代理集成创建 REST API
    - 创建 Fargate 启动类型的 Amazon ECS Linux 任务
    - 将函数名称作为变量创建 CloudWatch 控制面板
    - 创建 EC2 启动类型的 Amazon ECS 服务
    - 创建 Amazon Managed Grafana 工作区
    - 使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌
    - 开始使用 Amazon ECR
    - 开始使用 Amazon EKS
    - Amazon MSK 入门
    - Amazon OpenSearch Service 入门
    - Amazon SageMaker Feature Store 入门
    - Config 入门
    - 终端用户消息发送推送服务入门
    - IoT Core 入门
    - WAF 入门
    - 将硬编码的密钥移动到 Secrets Manager
    - 使用 FIS 在 EC2 实例上运行 CPU 压力测试
    - 设置 Systems Manager

使用 Amazon 开发工具包的 IAM 代码示例

以下代码示例显示如何将 IAM 与 Amazon 软件开发工具包（SDK）一起使用。

基础知识是向您展示如何在服务中执行基本操作的代码示例。

~~操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可~~以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务 服务结合来完成特定任务的代码示例。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

IAM

- 使用 Amazon SDK 的 IAM 基础知识示例
  - 开始使用 IAM
  - 了解将 IAM 与 Amazon SDK 结合使用的基础知识
  - 使用 Amazon 开发工具包的 IAM 操作
    - 将 AddClientIdToOpenIdConnectProvider 与 CLI 配合使用
    - 将 AddRoleToInstanceProfile 与 CLI 配合使用
    - 将 AddUserToGroup 与 CLI 配合使用
    - 将 AttachGroupPolicy 与 CLI 配合使用
    - 将 AttachRolePolicy 与 Amazon SDK 或 CLI 配合使用
    - 将 AttachUserPolicy 与 Amazon SDK 或 CLI 配合使用
    - 将 ChangePassword 与 CLI 配合使用
    - 将 CreateAccessKey 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateAccountAlias 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateGroup 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateInstanceProfile 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateLoginProfile 与 CLI 配合使用
    - 将 CreateOpenIdConnectProvider 与 CLI 配合使用
    - 将 CreatePolicy 与 Amazon SDK 或 CLI 配合使用
    - 将 CreatePolicyVersion 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateRole 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateSAMLProvider 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateUser 与 Amazon SDK 或 CLI 配合使用
    - 将 CreateVirtualMfaDevice 与 CLI 配合使用
    - 将 DeactivateMfaDevice 与 CLI 配合使用

- 将 DeleteAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountPasswordPolicy 与 CLI 配合使用
- 将 DeleteGroup 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteGroupPolicy 与 CLI 配合使用
- 将 DeleteInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteLoginProfile 与 CLI 配合使用
- 将 DeleteOpenIdConnectProvider 与 CLI 配合使用
- 将 DeletePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeletePolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRolePermissionsBoundary 与 CLI 配合使用
- 将 DeleteRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSigningCertificate 与 CLI 配合使用
- 将 DeleteUser 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteUserPermissionsBoundary 与 CLI 配合使用
- 将 DeleteUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteVirtualMfaDevice 与 CLI 配合使用
- 将 DetachGroupPolicy 与 CLI 配合使用
- 将 DetachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DetachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 EnableMfaDevice 与 CLI 配合使用
- 将 GenerateCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GenerateServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetAccessKeyLastUsed 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountAuthorizationDetails 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountPasswordPolicy 与 Amazon SDK 或 CLI 配合使用

- 将 GetAccountSummary 与 Amazon SDK 或 CLI 配合使用
- 将 GetContextKeysForCustomPolicy 与 CLI 配合使用
- 将 GetContextKeysForPrincipalPolicy 与 CLI 配合使用
- 将 GetCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GetGroup 与 CLI 配合使用
- 将 GetGroupPolicy 与 CLI 配合使用
- 将 GetInstanceProfile 与 CLI 配合使用
- 将 GetLoginProfile 与 CLI 配合使用
- 将 GetOpenIdConnectProvider 与 CLI 配合使用
- 将 GetPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 GetRole 与 Amazon SDK 或 CLI 配合使用
- 将 GetRolePolicy 与 CLI 配合使用
- 将 GetSamlProvider 与 CLI 配合使用
- 将 GetServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 GetServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetServiceLastAccessedDetailsWithEntities 与 CLI 配合使用
- 将 GetServiceLinkedRoleDeletionStatus 与 Amazon SDK 或 CLI 配合使用
- 将 GetUser 与 Amazon SDK 或 CLI 配合使用
- 将 GetUserPolicy 与 CLI 配合使用
- 将 ListAccessKeys 与 Amazon SDK 或 CLI 配合使用
- 将 ListAccountAliases 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedGroupPolicies 与 CLI 配合使用
- 将 ListAttachedRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedUserPolicies 与 CLI 配合使用
- 将 ListEntitiesForPolicy 与 CLI 配合使用
- 将 ListGroupPolicies 与 CLI 配合使用
- 将 ListGroups 与 Amazon SDK 或 CLI 配合使用
- 将 ListGroupsForUser 与 CLI 配合使用
- 将 ListInstanceProfiles 与 CLI 配合使用

- 将 ListInstanceProfilesForRole 与 CLI 配合使用
- 将 ListMfaDevices 与 CLI 配合使用
- 将 ListOpenIdConnectProviders 与 CLI 配合使用
- 将 ListPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListPolicyVersions 与 Amazon SDK 或 CLI 配合使用
- 将 ListRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListRoleTags 与 CLI 配合使用
- 将 ListRoles 与 Amazon SDK 或 CLI 配合使用
- 将 ListSAMLProviders 与 Amazon SDK 或 CLI 配合使用
- 将 ListServerCertificates 与 Amazon SDK 或 CLI 配合使用
- 将 ListSigningCertificates 与 CLI 配合使用
- 将 ListUserPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListUserTags 与 CLI 配合使用
- 将 ListUsers 与 Amazon SDK 或 CLI 配合使用
- 将 ListVirtualMfaDevices 与 CLI 配合使用
- 将 PutGroupPolicy 与 CLI 配合使用
- 将 PutRolePermissionsBoundary 与 CLI 配合使用
- 将 PutRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 PutUserPermissionsBoundary 与 CLI 配合使用
- 将 PutUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 RemoveClientIdFromOpenIdConnectProvider 与 CLI 配合使用
- 将 RemoveRoleFromInstanceProfile 与 CLI 配合使用
- 将 RemoveUserFromGroup 与 CLI 配合使用
- 将 ResyncMfaDevice 与 CLI 配合使用
- 将 SetDefaultPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 TagRole 与 CLI 配合使用
- 将 TagUser 与 CLI 配合使用
- 将 UntagRole 与 CLI 配合使用
- 将 UntagUser 与 CLI 配合使用
- 将 UpdateAccessKey 与 Amazon SDK 或 CLI 配合使用

    - 将 UpdateAccountPasswordPolicy 与 CLI 配合使用
    - 将 UpdateAssumeRolePolicy 与 CLI 配合使用
    - 将 UpdateGroup 与 CLI 配合使用
    - 将 UpdateLoginProfile 与 CLI 配合使用
    - 将 UpdateOpenIdConnectProviderThumbprint 与 CLI 配合使用
    - 将 UpdateRole 与 CLI 配合使用
    - 将 UpdateRoleDescription 与 CLI 配合使用
    - 将 UpdateSamlProvider 与 CLI 配合使用
    - 将 UpdateServerCertificate 与 Amazon SDK 或 CLI 配合使用
    - 将 UpdateSigningCertificate 与 CLI 配合使用
    - 将 UpdateUser 与 Amazon SDK 或 CLI 配合使用
    - 将 UploadServerCertificate 与 Amazon SDK 或 CLI 配合使用
    - 将 UploadSigningCertificate 与 CLI 配合使用
- 使用 Amazon 开发工具包的 IAM 场景
  - 使用 Amazon 开发工具包构建和管理弹性服务
  - 配置 Amazon ECS Service Connect
  - 使用 Lambda 代理集成创建 REST API
  - 创建 Fargate 启动类型的 Amazon ECS Linux 任务
  - 使用 Amazon 开发工具包创建只读和读写 IAM 用户
  - 将函数名称作为变量创建 CloudWatch 控制面板
  - 创建 EC2 启动类型的 Amazon ECS 服务
  - 创建 Amazon Managed Grafana 工作区
  - 创建第一个 Lambda 函数
  - 通过 CLI 开始使用 Redshift Serverless
  - IoT Device Defender 入门
  - 开始使用 Amazon EKS
  - Amazon MSK 入门
  - Amazon Redshift 预置集群入门
  - Amazon SageMaker Feature Store 入门
  - Config 入门

- Step Functions 入门
- 使用 Amazon 开发工具包管理 IAM 访问密钥
- 使用 Amazon 开发工具包管理 IAM policy
- 使用 Amazon 开发工具包管理 IAM 角色
- 使用 Amazon 开发工具包管理您的 IAM 账户
- 将硬编码的密钥移动到 Secrets Manager
- 允许 Amazon Compute Optimizer 自动化功能应用建议的操作
- 在整个组织中启用自动化功能的策略
- 启用账户自动化功能的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 授予计算优化自动化功能的服务相关角色权限的策略
- 使用 Amazon SDK 回滚 IAM policy 版本
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 使用 Amazon Command Line Interface v2 为 DynamoDB 设置基于属性的访问控制
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数
- 使用 Amazon Command Line Interface v2 处理 DynamoDB Streams 和生存时间
- 通过 Amazon SDK 使用 IAM Policy Builder API

使用 Amazon SDK 的 IAM 基础知识示例

以下代码示例展示了如何将 Amazon Identity and Access Management 的基本功能与 Amazon SDK 结合使用。

示例

- 开始使用 IAM
- 了解将 IAM 与 Amazon SDK 结合使用的基础知识
- 使用 Amazon 开发工具包的 IAM 操作
  - 将 AddClientIdToOpenIdConnectProvider 与 CLI 配合使用

- 将 AddRoleToInstanceProfile 与 CLI 配合使用
- 将 AddUserToGroup 与 CLI 配合使用
- 将 AttachGroupPolicy 与 CLI 配合使用
- 将 AttachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 AttachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 ChangePassword 与 CLI 配合使用
- 将 CreateAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 CreateAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 CreateGroup 与 Amazon SDK 或 CLI 配合使用
- 将 CreateInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 CreateLoginProfile 与 CLI 配合使用
- 将 CreateOpenIdConnectProvider 与 CLI 配合使用
- 将 CreatePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 CreatePolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 CreateRole 与 Amazon SDK 或 CLI 配合使用
- 将 CreateSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 CreateServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 CreateUser 与 Amazon SDK 或 CLI 配合使用
- 将 CreateVirtualMfaDevice 与 CLI 配合使用
- 将 DeactivateMfaDevice 与 CLI 配合使用
- 将 DeleteAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountPasswordPolicy 与 CLI 配合使用
- 将 DeleteGroup 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteGroupPolicy 与 CLI 配合使用
- 将 DeleteInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteLoginProfile 与 CLI 配合使用
- 将 DeleteOpenIdConnectProvider 与 CLI 配合使用
- 将 DeletePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeletePolicyVersion 与 Amazon SDK 或 CLI 配合使用

- 将 DeleteRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRolePermissionsBoundary 与 CLI 配合使用
- 将 DeleteRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSigningCertificate 与 CLI 配合使用
- 将 DeleteUser 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteUserPermissionsBoundary 与 CLI 配合使用
- 将 DeleteUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteVirtualMfaDevice 与 CLI 配合使用
- 将 DetachGroupPolicy 与 CLI 配合使用
- 将 DetachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DetachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 EnableMfaDevice 与 CLI 配合使用
- 将 GenerateCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GenerateServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetAccessKeyLastUsed 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountAuthorizationDetails 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountPasswordPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountSummary 与 Amazon SDK 或 CLI 配合使用
- 将 GetContextKeysForCustomPolicy 与 CLI 配合使用
- 将 GetContextKeysForPrincipalPolicy 与 CLI 配合使用
- 将 GetCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GetGroup 与 CLI 配合使用
- 将 GetGroupPolicy 与 CLI 配合使用
- 将 GetInstanceProfile 与 CLI 配合使用
- 将 GetLoginProfile 与 CLI 配合使用
- 将 GetOpenIdConnectProvider 与 CLI 配合使用
- 将 GetPolicy 与 Amazon SDK 或 CLI 配合使用

- 将 GetPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 GetRole 与 Amazon SDK 或 CLI 配合使用
- 将 GetRolePolicy 与 CLI 配合使用
- 将 GetSamlProvider 与 CLI 配合使用
- 将 GetServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 GetServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetServiceLastAccessedDetailsWithEntities 与 CLI 配合使用
- 将 GetServiceLinkedRoleDeletionStatus 与 Amazon SDK 或 CLI 配合使用
- 将 GetUser 与 Amazon SDK 或 CLI 配合使用
- 将 GetUserPolicy 与 CLI 配合使用
- 将 ListAccessKeys 与 Amazon SDK 或 CLI 配合使用
- 将 ListAccountAliases 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedGroupPolicies 与 CLI 配合使用
- 将 ListAttachedRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedUserPolicies 与 CLI 配合使用
- 将 ListEntitiesForPolicy 与 CLI 配合使用
- 将 ListGroupPolicies 与 CLI 配合使用
- 将 ListGroups 与 Amazon SDK 或 CLI 配合使用
- 将 ListGroupsForUser 与 CLI 配合使用
- 将 ListInstanceProfiles 与 CLI 配合使用
- 将 ListInstanceProfilesForRole 与 CLI 配合使用
- 将 ListMfaDevices 与 CLI 配合使用
- 将 ListOpenIdConnectProviders 与 CLI 配合使用
- 将 ListPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListPolicyVersions 与 Amazon SDK 或 CLI 配合使用
- 将 ListRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListRoleTags 与 CLI 配合使用
- 将 ListRoles 与 Amazon SDK 或 CLI 配合使用
- 将 ListSAMLProviders 与 Amazon SDK 或 CLI 配合使用
- 将 ListServerCertificates 与 Amazon SDK 或 CLI 配合使用

- 将 ListSigningCertificates 与 CLI 配合使用
- 将 ListUserPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListUserTags 与 CLI 配合使用
- 将 ListUsers 与 Amazon SDK 或 CLI 配合使用
- 将 ListVirtualMfaDevices 与 CLI 配合使用
- 将 PutGroupPolicy 与 CLI 配合使用
- 将 PutRolePermissionsBoundary 与 CLI 配合使用
- 将 PutRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 PutUserPermissionsBoundary 与 CLI 配合使用
- 将 PutUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 RemoveClientIdFromOpenIdConnectProvider 与 CLI 配合使用
- 将 RemoveRoleFromInstanceProfile 与 CLI 配合使用
- 将 RemoveUserFromGroup 与 CLI 配合使用
- 将 ResyncMfaDevice 与 CLI 配合使用
- 将 SetDefaultPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 TagRole 与 CLI 配合使用
- 将 TagUser 与 CLI 配合使用
- 将 UntagRole 与 CLI 配合使用
- 将 UntagUser 与 CLI 配合使用
- 将 UpdateAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 UpdateAccountPasswordPolicy 与 CLI 配合使用
- 将 UpdateAssumeRolePolicy 与 CLI 配合使用
- 将 UpdateGroup 与 CLI 配合使用
- 将 UpdateLoginProfile 与 CLI 配合使用
- 将 UpdateOpenIdConnectProviderThumbprint 与 CLI 配合使用
- 将 UpdateRole 与 CLI 配合使用
- 将 UpdateRoleDescription 与 CLI 配合使用
- 将 UpdateSamlProvider 与 CLI 配合使用
- 将 UpdateServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UpdateSigningCertificate 与 CLI 配合使用

- 将 UpdateUser 与 Amazon SDK 或 CLI 配合使用
- 将 UploadServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UploadSigningCertificate 与 CLI 配合使用

开始使用 IAM

以下代码示例展示了如何开始使用 IAM。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
namespace IAMActions;

public class HelloIAM
{
    static async Task Main(string[] args)
    {
        // Getting started with AWS Identity and Access Management (IAM). List
        // the policies for the account.
        var iamClient = new AmazonIdentityManagementServiceClient();

        var listPoliciesPaginator = iamClient.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        Console.WriteLine("Here are the policies defined for your account:\n");
        policies.ForEach(policy =>
        {
```

```
            Console.WriteLine($"Created:
 {policy.CreateDate}\t{policy.PolicyName}\t{policy.Description}");
        });
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 ListPolicies。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS iam)

# Set this project's name.
project("hello_iam")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
```

```
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
    # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

    # set(BIN_SUB_DIR "/Debug") # if you are building from the command line you
 may need to uncomment this
    # and set the proper subdirectory to the executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_iam.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

iam.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/iam/IAMClient.h>
#include <aws/iam/model/ListPoliciesRequest.h>
#include <iostream>
#include <iomanip>

/*
 *  A "Hello IAM" starter application which initializes an AWS Identity and
 Access Management (IAM) client
 *  and lists the IAM policies.
 *
 *  main function
 *
 *  Usage: 'hello_iam'
```

```
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        const Aws::String DATE_FORMAT("%Y-%m-%d");
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::IAM::IAMClient iamClient(clientConfig);
        Aws::IAM::Model::ListPoliciesRequest request;

        bool done = false;
        bool header = false;
        while (!done) {
            auto outcome = iamClient.ListPolicies(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Failed to list iam policies: " <<
                          outcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }

            if (!header) {
                std::cout << std::left << std::setw(55) << "Name" <<
                          std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                          std::setw(64) << "Description" << std::setw(12) <<
                          "CreateDate" << std::endl;
                header = true;
            }

            const auto &policies = outcome.GetResult().GetPolicies();
            for (const auto &policy: policies) {
                std::cout << std::left << std::setw(55) <<
                          policy.GetPolicyName() << std::setw(30) <<
                          policy.GetPolicyId() << std::setw(80) <<
 policy.GetArn() <<
```

```
                            std::setw(64) << policy.GetDescription() <<
 std::setw(12) <<
                            policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
 <<
                            std::endl;
            }

            if (outcome.GetResult().GetIsTruncated()) {
                request.SetMarker(outcome.GetResult().GetMarker());
            } else {
                done = true;
            }
        }
    }


    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 ListPolicies。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
package main

import (
 "context"
 "fmt"
```

```
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/iam"
)

// main uses the AWS SDK for Go (v2) to create an AWS Identity and Access
 Management (IAM)
// client and list up to 10 policies in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 iamClient := iam.NewFromConfig(sdkConfig)
 const maxPols = 10
 fmt.Printf("Let's list up to %v policies for your account.\n", maxPols)
 result, err := iamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPols),
 })
 if err != nil {
  fmt.Printf("Couldn't list policies for your account. Here's why: %v\n", err)
  return
 }
 if len(result.Policies) == 0 {
  fmt.Println("You don't have any policies!")
 } else {
  for _, policy := range result.Policies {
   fmt.Printf("\t%v\n", *policy.PolicyName)
  }
 }
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 ListPolicies。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.ListPoliciesResponse;
import software.amazon.awssdk.services.iam.model.Policy;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class HelloIAM {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listPolicies(iam);
    }

    public static void listPolicies(IamClient iam) {
        ListPoliciesResponse response = iam.listPolicies();
        List<Policy> polList = response.policies();
        polList.forEach(policy -> {
            System.out.println("Policy Name: " + policy.policyName());
        });
    }
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 ListPolicies。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { IAMClient, paginateListPolicies } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listLocalPolicies = async () => {
  /**
   * In v3, the clients expose paginateOperationName APIs that are written using
 async generators so that you can use async iterators in a for await..of loop.
   * https://docs.aws.amazon.com/AWSJavaScriptSDK/v3/latest/index.html#paginators
   */
  const paginator = paginateListPolicies(
    { client, pageSize: 10 },
    // List only customer managed policies.
    { Scope: "Local" },
  );

  console.log("IAM policies defined in your account:");
  let policyCount = 0;
  for await (const page of paginator) {
    if (page.Policies) {
      for (const policy of page.Policies) {
        console.log(`${policy.PolicyName}`);
        policyCount++;
      }
    }
  }
  console.log(`Found ${policyCount} policies.`);
```

```
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListPolicies。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import boto3


def main():
    """
    Lists the managed policies in your AWS account using the AWS SDK for Python
 (Boto3).
    """
    iam = boto3.client("iam")

    try:
        # Get a paginator for the list_policies operation
        paginator = iam.get_paginator("list_policies")

        # Iterate through the pages of results
        for page in paginator.paginate(Scope="All", OnlyAttached=False):
            for policy in page["Policies"]:
                print(f"Policy name: {policy['PolicyName']}")
                print(f"  Policy ARN: {policy['Arn']}")
    except boto3.exceptions.BotoCoreError as e:
        print(f"Encountered an error while listing policies: {e}")


if __name__ == "__main__":
```

```
    main()
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 ListPolicies。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-iam'
require 'logger'

# IAMManager is a class responsible for managing IAM operations
# such as listing all IAM policies in the current AWS account.
class IAMManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all IAM policies in the current AWS account.
  def list_policies
    @logger.info('Here are the IAM policies in your account:')

    paginator = @client.list_policies
    policies = []

    paginator.each_page do |page|
      policies.concat(page.policies)
    end

    if policies.empty?
```

```
      @logger.info("You don't have any IAM policies.")
    else
      policies.each do |policy|
        @logger.info("- #{policy.policy_name}")
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  iam_client = Aws::IAM::Client.new
  manager = IAMManager.new(iam_client)
  manager.list_policies
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 ListPolicies。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

来自 src/bin/hello.rs。

```
use aws_sdk_iam::error::SdkError;
use aws_sdk_iam::operation::list_policies::ListPoliciesError;
use clap::Parser;

const PATH_PREFIX_HELP: &str = "The path prefix for filtering the results.";

#[derive(Debug, clap::Parser)]
#[command(about)]
struct HelloScenarioArgs {
```

```
    #[arg(long, default_value="/", help=PATH_PREFIX_HELP)]
    pub path_prefix: String,
}

#[tokio::main]
async fn main() -> Result<(), SdkError<ListPoliciesError>> {
    let sdk_config = aws_config::load_from_env().await;
    let client = aws_sdk_iam::Client::new(&sdk_config);

    let args = HelloScenarioArgs::parse();

    iam_service::list_policies(client, args.path_prefix).await?;

    Ok(())
}
```

来自 src/iam-service-lib.rs。

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();
```

```
    Ok(policy_names)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

了解将 IAM 与 Amazon SDK 结合使用的基础知识

以下代码示例演示了如何创建用户并代入角色。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建没有权限的用户。
- 创建授予列出账户的 Amazon S3 存储桶的权限的角色
- 添加策略以允许用户代入该角色。
- 代入角色并使用临时凭证列出 S3 存储桶，然后清除资源。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
global using Amazon.IdentityManagement;
global using Amazon.S3;
global using Amazon.SecurityToken;
```

```
global using IAMActions;
global using IamScenariosCommon;
global using Microsoft.Extensions.DependencyInjection;
global using Microsoft.Extensions.Hosting;
global using Microsoft.Extensions.Logging;
global using Microsoft.Extensions.Logging.Console;
global using Microsoft.Extensions.Logging.Debug;


namespace IAMActions;

public class IAMWrapper
{
    private readonly IAmazonIdentityManagementService _IAMService;

    /// <summary>
    /// Constructor for the IAMWrapper class.
    /// </summary>
    /// <param name="IAMService">An IAM client object.</param>
    public IAMWrapper(IAmazonIdentityManagementService IAMService)
    {
        _IAMService = IAMService;
    }

    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }


    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }


    /// <summary>
    /// Create a new IAM role.
    /// </summary>
```

```
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
 rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }


    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }


    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
```

```
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
```

```
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
 { RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
 { UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
```

```
    }


    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Detach an IAM policy from an IAM role.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
    {
        var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }


    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
```

```
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
 GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }


    /// <summary>
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }


    /// <summary>
    /// Get information about an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to retrieve information
    /// for.</param>
    /// <returns>The IAM role that was retrieved.</returns>
    public async Task<Role> GetRoleAsync(string roleName)
    {
        var response = await _IAMService.GetRoleAsync(new GetRoleRequest
        {
            RoleName = roleName,
        });

        return response.Role;
    }


    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
```

```
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }


    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }


    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
```

```
        }

        return groups;
    }


    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
 ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }


    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
```

```
    }


    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }

        return roles;
    }


    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }


    /// <summary>
    /// List IAM users.
    /// </summary>
    /// <returns>A list of IAM users.</returns>
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
ListUsersRequest());
        var users = new List<User>();
```

```
        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }


    /// <summary>
    /// Add or update an inline policy document that is embedded in an IAM user.
    /// </summary>
    /// <param name="userName">The name of the IAM user.</param>
    /// <param name="policyName">The name of the IAM policy.</param>
    /// <param name="policyDocument">The policy document defining the IAM
 policy.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutUserPolicyAsync(string userName, string
 policyName, string policyDocument)
    {
        var request = new PutUserPolicyRequest
```

```
        {
            UserName = userName,
            PolicyName = policyName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutUserPolicyAsync(request);
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }

    /// <summary>
    /// Wait for a new access key to be ready to use.
    /// </summary>
    /// <param name="accessKeyId">The Id of the access key.</param>
    /// <returns>A boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitUntilAccessKeyIsReady(string accessKeyId)
    {
        var keyReady = false;

        do
        {
            try
            {
                var response = await _IAMService.GetAccessKeyLastUsedAsync(
                    new GetAccessKeyLastUsedRequest { AccessKeyId =
 accessKeyId });
                if (response.UserName is not null)
                {
                    keyReady = true;
                }
            }
            catch (NoSuchEntityException)
            {
                keyReady = false;
            }
        } while (!keyReady);

        return keyReady;
    }
}



using Microsoft.Extensions.Configuration;
```

```
namespace IAMBasics;

public class IAMBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for the AWS service.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonIdentityManagementService>()
            .AddTransient<IAMWrapper>()
            .AddTransient<UIWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<IAMBasics>();


        IConfiguration configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load test settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
            .Build();

        // Values needed for user, role, and policies.
        string userName = configuration["UserName"]!;
        string s3PolicyName = configuration["S3PolicyName"]!;
        string roleName = configuration["RoleName"]!;


        var iamWrapper = host.Services.GetRequiredService<IAMWrapper>();
        var uiWrapper = host.Services.GetRequiredService<UIWrapper>();
```

```
        uiWrapper.DisplayBasicsOverview();
        uiWrapper.PressEnter();

        // First create a user. By default, the new user has
        // no permissions.
        uiWrapper.DisplayTitle("Create User");
        Console.WriteLine($"Creating a new user with user name: {userName}.");
        var user = await iamWrapper.CreateUserAsync(userName);
        var userArn = user.Arn;

        Console.WriteLine($"Successfully created user: {userName} with ARN:
{userArn}.");
        uiWrapper.WaitABit(15, "Now let's wait for the user to be ready for
use.");

        // Define a role policy document that allows the new user
        // to assume the role.
        string assumeRolePolicyDocument = "{" +
          "\"Version\": \"2012-10-17\"," +
          "\"Statement\": [{" +
              "\"Effect\": \"Allow\"," +
              "\"Principal\": {" +
              $" \"AWS\": \"{userArn}\"" +
              "}," +
              "\"Action\": \"sts:AssumeRole\"" +
          "}]" +
        "}";

        // Permissions to list all buckets.
        string policyDocument = "{" +
            "\"Version\": \"2012-10-17\"," +
            " \"Statement\" : [{" +
                " \"Action\" : [\"s3:ListAllMyBuckets\"]," +
                " \"Effect\" : \"Allow\"," +
                " \"Resource\" : \"*\"" +
            "}]" +
        "}";

        // Create an AccessKey for the user.
        uiWrapper.DisplayTitle("Create access key");
        Console.WriteLine("Now let's create an access key for the new user.");
        var accessKey = await iamWrapper.CreateAccessKeyAsync(userName);

        var accessKeyId = accessKey.AccessKeyId;
```

```
        var secretAccessKey = accessKey.SecretAccessKey;

        Console.WriteLine($"We have created the access key with Access key id:
{accessKeyId}.");

        Console.WriteLine("Now let's wait until the IAM access key is ready to
use.");
        var keyReady = await iamWrapper.WaitUntilAccessKeyIsReady(accessKeyId);

        // Now try listing the Amazon Simple Storage Service (Amazon S3)
        // buckets. This should fail at this point because the user doesn't
        // have permissions to perform this task.
        uiWrapper.DisplayTitle("Try to display Amazon S3 buckets");
        Console.WriteLine("Now let's try to display a list of the user's Amazon
S3 buckets.");
        var s3Client1 = new AmazonS3Client(accessKeyId, secretAccessKey);
        var stsClient1 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        var s3Wrapper = new S3Wrapper(s3Client1, stsClient1);
        var buckets = await s3Wrapper.ListMyBucketsAsync();

        Console.WriteLine(buckets is null
            ? "As expected, the call to list the buckets has returned a null
list."
            : "Something went wrong. This shouldn't have worked.");

        uiWrapper.PressEnter();

        uiWrapper.DisplayTitle("Create IAM role");
        Console.WriteLine($"Creating the role: {roleName}");

        // Creating an IAM role to allow listing the S3 buckets. A role name
        // is not case sensitive and must be unique to the account for which it
        // is created.
        var roleArn = await iamWrapper.CreateRoleAsync(roleName,
assumeRolePolicyDocument);

        uiWrapper.PressEnter();

        // Create a policy with permissions to list S3 buckets.
        uiWrapper.DisplayTitle("Create IAM policy");
        Console.WriteLine($"Creating the policy: {s3PolicyName}");
```

```
        Console.WriteLine("with permissions to list the Amazon S3 buckets for the
account.");
        var policy = await iamWrapper.CreatePolicyAsync(s3PolicyName,
policyDocument);

        // Wait 15 seconds for the IAM policy to be available.
        uiWrapper.WaitABit(15, "Waiting for the policy to be available.");

        // Attach the policy to the role you created earlier.
        uiWrapper.DisplayTitle("Attach new IAM policy");
        Console.WriteLine("Now let's attach the policy to the role.");
        await iamWrapper.AttachRolePolicyAsync(policy.Arn, roleName);

        // Wait 15 seconds for the role to be updated.
        Console.WriteLine();
        uiWrapper.WaitABit(15, "Waiting for the policy to be attached.");

        // Use the AWS Security Token Service (AWS STS) to have the user
        // assume the role we created.
        var stsClient2 = new AmazonSecurityTokenServiceClient(accessKeyId,
secretAccessKey);

        // Wait for the new credentials to become valid.
        uiWrapper.WaitABit(10, "Waiting for the credentials to be valid.");

        var assumedRoleCredentials = await
s3Wrapper.AssumeS3RoleAsync("temporary-session", roleArn);

        // Try again to list the buckets using the client created with
        // the new user's credentials. This time, it should work.
        var s3Client2 = new AmazonS3Client(assumedRoleCredentials);

        s3Wrapper.UpdateClients(s3Client2, stsClient2);

        buckets = await s3Wrapper.ListMyBucketsAsync();

        uiWrapper.DisplayTitle("List Amazon S3 buckets");
        Console.WriteLine("This time we should have buckets to list.");
        if (buckets is not null)
        {
            buckets.ForEach(bucket =>
            {
                Console.WriteLine($"{bucket.BucketName} created:
{bucket.CreationDate}");
```

```
            });
        }

        uiWrapper.PressEnter();

        // Now clean up all the resources used in the example.
        uiWrapper.DisplayTitle("Clean up resources");
        Console.WriteLine("Thank you for watching. The IAM Basics demo is
 complete.");
        Console.WriteLine("Please wait while we clean up the resources we
 created.");

        await iamWrapper.DetachRolePolicyAsync(policy.Arn, roleName);

        await iamWrapper.DeletePolicyAsync(policy.Arn);

        await iamWrapper.DeleteRoleAsync(roleName);

        await iamWrapper.DeleteAccessKeyAsync(accessKeyId, userName);

        await iamWrapper.DeleteUserAsync(userName);

        uiWrapper.PressEnter();

        Console.WriteLine("All done cleaning up our resources. Thank you for your
 patience.");
    }
}


namespace IamScenariosCommon;

using System.Net;

/// <summary>
/// A class to perform Amazon Simple Storage Service (Amazon S3) actions for
/// the IAM Basics scenario.
/// </summary>
public class S3Wrapper
{
    private IAmazonS3 _s3Service;
    private IAmazonSecurityTokenService _stsService;

    /// <summary>
```

```
    /// Constructor for the S3Wrapper class.
    /// </summary>
    /// <param name="s3Service">An Amazon S3 client object.</param>
    /// <param name="stsService">An AWS Security Token Service (AWS STS)
    /// client object.</param>
    public S3Wrapper(IAmazonS3 s3Service, IAmazonSecurityTokenService stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }

    /// <summary>
    /// Assumes an AWS Identity and Access Management (IAM) role that allows
    /// Amazon S3 access for the current session.
    /// </summary>
    /// <param name="roleSession">A string representing the current session.</
param>
    /// <param name="roleToAssume">The name of the IAM role to assume.</param>
    /// <returns>Credentials for the newly assumed IAM role.</returns>
    public async Task<Credentials> AssumeS3RoleAsync(string roleSession, string
 roleToAssume)
    {
        // Create the request to use with the AssumeRoleAsync call.
        var request = new AssumeRoleRequest()
        {
            RoleSessionName = roleSession,
            RoleArn = roleToAssume,
        };

        var response = await _stsService.AssumeRoleAsync(request);

        return response.Credentials;
    }


    /// <summary>
    /// Delete an S3 bucket.
    /// </summary>
    /// <param name="bucketName">Name of the S3 bucket to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteBucketAsync(string bucketName)
    {
        var result = await _s3Service.DeleteBucketAsync(new DeleteBucketRequest
 { BucketName = bucketName });
```

```
        return result.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// List the buckets that are owned by the user's account.
    /// </summary>
    /// <returns>Async Task.</returns>
    public async Task<List<S3Bucket>?> ListMyBucketsAsync()
    {
        try
        {
            // Get the list of buckets accessible by the new user.
            var response = await _s3Service.ListBucketsAsync();

            return response.Buckets;
        }
        catch (AmazonS3Exception ex)
        {
            // Something else went wrong. Display the error message.
            Console.WriteLine($"Error: {ex.Message}");
            return null;
        }
    }

    /// <summary>
    /// Create a new S3 bucket.
    /// </summary>
    /// <param name="bucketName">The name for the new bucket.</param>
    /// <returns>A Boolean value indicating whether the action completed
    /// successfully.</returns>
    public async Task<bool> PutBucketAsync(string bucketName)
    {
        var response = await _s3Service.PutBucketAsync(new PutBucketRequest
{ BucketName = bucketName });
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

    /// <summary>
    /// Update the client objects with new client objects. This is available
    /// because the scenario uses the methods of this class without and then
    /// with the proper permissions to list S3 buckets.
    /// </summary>
    /// <param name="s3Service">The Amazon S3 client object.</param>
    /// <param name="stsService">The AWS STS client object.</param>
```

```
    public void UpdateClients(IAmazonS3 s3Service, IAmazonSecurityTokenService
 stsService)
    {
        _s3Service = s3Service;
        _stsService = stsService;
    }
}


namespace IamScenariosCommon;

public class UIWrapper
{
    public readonly string SepBar = new('-', Console.WindowWidth);

    /// <summary>
    /// Show information about the IAM Groups scenario.
    /// </summary>
    public void DisplayGroupsOverview()
    {
        Console.Clear();

        DisplayTitle("Welcome to the IAM Groups Demo");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates an Amazon Identity and Access Management
 (IAM) group.");
        Console.WriteLine("\t2. Adds an IAM policy to the IAM group giving it
 full access to Amazon S3.");
        Console.WriteLine("\t3. Creates a new IAM user.");
        Console.WriteLine("\t4. Creates an IAM access key for the user.");
        Console.WriteLine("\t5. Adds the user to the IAM group.");
        Console.WriteLine("\t6. Lists the buckets on the account.");
        Console.WriteLine("\t7. Proves that the user has full Amazon S3 access by
 creating a bucket.");
        Console.WriteLine("\t8. List the buckets again to show the new bucket.");
        Console.WriteLine("\t9. Cleans up all the resources created.");
    }

    /// <summary>
    /// Show information about the IAM Basics scenario.
    /// </summary>
    public void DisplayBasicsOverview()
    {
        Console.Clear();
```

```
        DisplayTitle("Welcome to IAM Basics");
        Console.WriteLine("This example application does the following:");
        Console.WriteLine("\t1. Creates a user with no permissions.");
        Console.WriteLine("\t2. Creates a role and policy that grant
s3:ListAllMyBuckets permission.");
        Console.WriteLine("\t3. Grants the user permission to assume the role.");
        Console.WriteLine("\t4. Creates an S3 client object as the user and tries
to list buckets (this will fail).");
        Console.WriteLine("\t5. Gets temporary credentials by assuming the
role.");
        Console.WriteLine("\t6. Creates a new S3 client object with the temporary
credentials and lists the buckets (this will succeed).");
        Console.WriteLine("\t7. Deletes all the resources.");
    }

    /// <summary>
    /// Display a message and wait until the user presses enter.
    /// </summary>
    public void PressEnter()
    {
        Console.Write("\nPress <Enter> to continue. ");
        _ = Console.ReadLine();
        Console.WriteLine();
    }

    /// <summary>
    /// Pad a string with spaces to center it on the console display.
    /// </summary>
    /// <param name="strToCenter">The string to be centered.</param>
    /// <returns>The padded string.</returns>
    public string CenterString(string strToCenter)
    {
        var padAmount = (Console.WindowWidth - strToCenter.Length) / 2;
        var leftPad = new string(' ', padAmount);
        return $"{leftPad}{strToCenter}";
    }

    /// <summary>
    /// Display a line of hyphens, the centered text of the title, and another
    /// line of hyphens.
    /// </summary>
    /// <param name="strTitle">The string to be displayed.</param>
    public void DisplayTitle(string strTitle)
```

```
    {
        Console.WriteLine(SepBar);
        Console.WriteLine(CenterString(strTitle));
        Console.WriteLine(SepBar);
    }

    /// <summary>
    /// Display a countdown and wait for a number of seconds.
    /// </summary>
    /// <param name="numSeconds">The number of seconds to wait.</param>
    public void WaitABit(int numSeconds, string msg)
    {
        Console.WriteLine(msg);

        // Wait for the requested number of seconds.
        for (int i = numSeconds; i > 0; i--)
        {
            System.Threading.Thread.Sleep(1000);
            Console.Write($"{i}...");
        }

        PressEnter();
    }
}
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy

- PutUserPolicy

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
#############################################################################
# function iam_create_user_assume_role
#
# Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
#
#     "IAM access" permissions are needed to run this code.
#     "STS assume role" permissions are needed to run this code. (Note: It might
 be necessary to
#           create a custom policy).
#
# Returns:
#       0 - If successful.
#       1 - If an error occurred.
#############################################################################
function iam_create_user_assume_role() {
  {
    if [ "$IAM_OPERATIONS_SOURCED" != "True" ]; then

      source ./iam_operations.sh
    fi
  }

  echo_repeat "*" 88
  echo "Welcome to the IAM create user and assume role demo."
  echo
  echo "This demo will create an IAM user, create an IAM role, and apply the role
 to the user."
  echo_repeat "*" 88
```

```
  echo

  echo -n "Enter a name for a new IAM user: "
  get_input
  user_name=$get_input_result

  local user_arn
  user_arn=$(iam_create_user -u "$user_name")

  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created demo IAM user named $user_name"
  else
    errecho "$user_arn"
    errecho "The user failed to create. This demo will exit."
    return 1
  fi

  local access_key_response
  access_key_response=$(iam_create_user_access_key -u "$user_name")
  # shellcheck disable=SC2181
  if [[ ${?} != 0 ]]; then
    errecho "The access key failed to create. This demo will exit."
    clean_up "$user_name"
    return 1
  fi

  IFS=$'\t ' read -r -a access_key_values <<<"$access_key_response"
  local key_name=${access_key_values[0]}
  local key_secret=${access_key_values[1]}

  echo "Created access key named $key_name"

  echo "Wait 10 seconds for the user to be ready."
  sleep 10
  echo_repeat "*" 88
  echo

  local iam_role_name
  iam_role_name=$(generate_random_name "test-role")
  echo "Creating a role named $iam_role_name with user $user_name as the
principal."

  local assume_role_policy_document="{
```

```
    \"Version\": \"2012-10-17\",
    \"Statement\": [{
        \"Effect\": \"Allow\",
        \"Principal\": {\"AWS\": \"$user_arn\"},
        \"Action\": \"sts:AssumeRole\"
        }]
    }"

  local role_arn
  role_arn=$(iam_create_role -n "$iam_role_name" -p
"$assume_role_policy_document")

  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created IAM role named $iam_role_name"
  else
    errecho "The role failed to create. This demo will exit."
    clean_up "$user_name" "$key_name"
    return 1
  fi

  local policy_name
  policy_name=$(generate_random_name "test-policy")
  local policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]}"

  local policy_arn
  policy_arn=$(iam_create_policy -n "$policy_name" -p "$policy_document")
  # shellcheck disable=SC2181
  if [[ ${?} == 0 ]]; then
    echo "Created  IAM policy named $policy_name"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name"
    return 1
  fi

  if (iam_attach_role_policy -n "$iam_role_name" -p "$policy_arn"); then
    echo "Attached policy $policy_arn to role $iam_role_name"
  else
```

```
    errecho "The policy failed to attach."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
    return 1
  fi

  local assume_role_policy_document="{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"$role_arn\"}]}"

  local assume_role_policy_name
  assume_role_policy_name=$(generate_random_name "test-assume-role-")

  # shellcheck disable=SC2181
  local assume_role_policy_arn
  assume_role_policy_arn=$(iam_create_policy -n "$assume_role_policy_name" -p
"$assume_role_policy_document")
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    echo "Created  IAM policy named $assume_role_policy_name for sts assume role"
  else
    errecho "The policy failed to create."
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn"
    return 1
  fi

  echo "Wait 10 seconds to give AWS time to propagate these new resources and
connections."
  sleep 10
  echo_repeat "*" 88
  echo

  echo "Try to list buckets without the new user assuming the role."
  echo_repeat "*" 88
  echo

  # Set the environment variables for the created user.
  # bashsupport disable=BP2001
  export AWS_ACCESS_KEY_ID=$key_name
  # bashsupport disable=BP2001
  export AWS_SECRET_ACCESS_KEY=$key_secret
```

```
 local buckets
 buckets=$(s3_list_buckets)

 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   local bucket_count
   bucket_count=$(echo "$buckets" | wc -w | xargs)
   echo "There are $bucket_count buckets in the account. This should not have
happened."
 else
   errecho "Because the role with permissions has not been assumed, listing
buckets failed."
 fi

 echo
 echo_repeat "*" 88
 echo "Now assume the role $iam_role_name and list the buckets."
 echo_repeat "*" 88
 echo

 local credentials

 credentials=$(sts_assume_role -r "$role_arn" -n "AssumeRoleDemoSession")
 # shellcheck disable=SC2181
 if [ ${?} == 0 ]; then
   echo "Assumed role $iam_role_name"
 else
   errecho "Failed to assume role."
   export AWS_ACCESS_KEY_ID=""
   export AWS_SECRET_ACCESS_KEY=""
   clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
"$policy_arn" "$assume_role_policy_arn"
   return 1
 fi

 IFS=$'\t ' read -r -a credentials <<<"$credentials"

 export AWS_ACCESS_KEY_ID=${credentials[0]}
 export AWS_SECRET_ACCESS_KEY=${credentials[1]}
 # bashsupport disable=BP2001
 export AWS_SESSION_TOKEN=${credentials[2]}

 buckets=$(s3_list_buckets)
```

```
  # shellcheck disable=SC2181
  if [ ${?} == 0 ]; then
    local bucket_count
    bucket_count=$(echo "$buckets" | wc -w | xargs)
    echo "There are $bucket_count buckets in the account. Listing buckets
 succeeded because of "
    echo "the assumed role."
  else
    errecho "Failed to list buckets. This should not happen."
    export AWS_ACCESS_KEY_ID=""
    export AWS_SECRET_ACCESS_KEY=""
    export AWS_SESSION_TOKEN=""
    clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn"
 "$policy_arn" "$assume_role_policy_arn"
    return 1
  fi

  local result=0
  export AWS_ACCESS_KEY_ID=""
  export AWS_SECRET_ACCESS_KEY=""

  echo
  echo_repeat "*" 88
  echo "The created resources will now be deleted."
  echo_repeat "*" 88
  echo

  clean_up "$user_name" "$key_name" "$iam_role_name" "$policy_arn" "$policy_arn"
 "$assume_role_policy_arn"

  # shellcheck disable=SC2181
  if [[ ${?} -ne 0 ]]; then
    result=1
  fi

  return $result
}
```

此场景中使用的 IAM 函数。

```
##############################################################################
```

```
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
```

```
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi
```

```
  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user already exists, we don't want to try to create it.
  if (iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name already exists in the account."
    return 1
  fi

  response=$(aws iam create-user --user-name "$user_name" \
    --output text \
    --query 'User.Arn')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-user operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
```

```
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
```

```
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}

###############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_json -- The assume role policy document."
```

```
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-role operation failed.\n$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
```

```
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}

###############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
```

```
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi
```

```
  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam attach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_detach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_detach_role_policy"
```

```
    echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
    echo "  -n role_name   The name of the IAM role."
    echo "  -p policy_ARN -- The IAM policy document ARN."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      p) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy ARN with the -p parameter."
    usage
    return 1
  fi

  response=$(aws iam detach-role-policy \
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}

###############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) policy_arn="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
```

```
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_arn" ]]; then
    errecho "ERROR: You must provide a policy arn with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Policy arn:  $policy_arn"
  iecho ""

  response=$(aws iam delete-policy \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
    return 1
  fi

  iecho "delete-policy response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
```

```
#       1 - If it fails.
###############################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_access_key"
    echo "Deletes an AWS Identity and Access Management (IAM) access key for the
 specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to delete."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:k:h" option; do
```

```
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    k) access_key="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

if [[ -z "$access_key" ]]; then
  errecho "ERROR: You must provide an access key with the -k parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Username:   $user_name"
iecho "    Access key:   $access_key"
iecho ""

response=$(aws iam delete-access-key \
  --user-name "$user_name" \
  --access-key-id "$access_key")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
  return 1
fi
```

```
  iecho "delete-access-key response:$response"
  iecho

  return 0
}

###############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
```

```
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey

- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
namespace AwsDoc {
    namespace IAM {

        //! Cleanup by deleting created entities.
        /*!
          \sa DeleteCreatedEntities
          \param client: IAM client.
          \param role: IAM role.
          \param user: IAM user.
          \param policy: IAM policy.
        */
        static bool DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
                                          const Aws::IAM::Model::Role &role,
                                          const Aws::IAM::Model::User &user,
                                          const Aws::IAM::Model::Policy &policy);
    }
```

```
    static const int LIST_BUCKETS_WAIT_SEC = 20;

    static const char ALLOCATION_TAG[] = "example_code";
}

//! Scenario to create an IAM user, create an IAM role, and apply the role to the
 user.
// "IAM access" permissions are needed to run this code.
// "STS assume role" permissions are needed to run this code. (Note: It might be
 necessary to
//    create a custom policy).
/*!
  \sa iamCreateUserAssumeRoleScenario
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
*/
bool AwsDoc::IAM::iamCreateUserAssumeRoleScenario(
        const Aws::Client::ClientConfiguration &clientConfig) {

    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::User user;
    Aws::IAM::Model::Role role;
    Aws::IAM::Model::Policy policy;

    // 1. Create a user.
    {
        Aws::IAM::Model::CreateUserRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String userName = "iam-demo-user-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetUserName(userName);

        Aws::IAM::Model::CreateUserOutcome outcome = client.CreateUser(request);
        if (!outcome.IsSuccess()) {
            std::cout << "Error creating IAM user " << userName << ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }
        else {
            std::cout << "Successfully created IAM user " << userName <<
 std::endl;
        }
```

```
        user = outcome.GetResult().GetUser();
    }

    // 2. Create a role.
    {
        // Get the IAM user for the current client in order to access its ARN.
        Aws::String iamUserArn;
        {
            Aws::IAM::Model::GetUserRequest request;
            Aws::IAM::Model::GetUserOutcome outcome = client.GetUser(request);
            if (!outcome.IsSuccess()) {
                std::cerr << "Error getting Iam user. " <<
                          outcome.GetError().GetMessage() << std::endl;

                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }
            else {
                std::cout << "Successfully retrieved Iam user "
                          << outcome.GetResult().GetUser().GetUserName()
                          << std::endl;
            }

            iamUserArn = outcome.GetResult().GetUser().GetArn();
        }

        Aws::IAM::Model::CreateRoleRequest request;

        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleName = "iam-demo-role-" +
                               Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleName(roleName);

        // Build policy document for role.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");

        Aws::Utils::Document jsonPrincipal;
        jsonPrincipal.WithString("AWS", iamUserArn);
        jsonStatement.WithObject("Principal", jsonPrincipal);
        jsonStatement.WithString("Action", "sts:AssumeRole");
        jsonStatement.WithObject("Condition", Aws::Utils::Document());

        Aws::Utils::Document policyDocument;
```

```
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Setting policy for role\n   "
                  << policyDocument.View().WriteCompact() << std::endl;

        // Set role policy document as JSON string.

request.SetAssumeRolePolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating role. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a role with name " << roleName
                      << std::endl;
        }

        role = outcome.GetResult().GetRole();
    }

    // 3. Create an IAM policy.
    {
        Aws::IAM::Model::CreatePolicyRequest request;
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String policyName = "iam-demo-policy-" +
                                 Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetPolicyName(policyName);

        // Build IAM policy document.
        Aws::Utils::Document jsonStatement;
        jsonStatement.WithString("Effect", "Allow");
        jsonStatement.WithString("Action", "s3:ListAllMyBuckets");
        jsonStatement.WithString("Resource", "arn:aws:s3:::*");

        Aws::Utils::Document policyDocument;
```

```
        policyDocument.WithString("Version", "2012-10-17");

        Aws::Utils::Array<Aws::Utils::Document> statements(1);
        statements[0] = jsonStatement;
        policyDocument.WithArray("Statement", statements);

        std::cout << "Creating a policy.\n   " <<
policyDocument.View().WriteCompact()
                  << std::endl;

        // Set IAM policy document as JSON string.
        request.SetPolicyDocument(policyDocument.View().WriteCompact());

        Aws::IAM::Model::CreatePolicyOutcome outcome =
client.CreatePolicy(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully created a policy with name, " <<
policyName <<
                      "." << std::endl;
        }

        policy = outcome.GetResult().GetPolicy();
    }

    // 4. Assume the new role using the AWS Security Token Service (STS).
    Aws::STS::Model::Credentials credentials;
    {
        Aws::STS::STSClient stsClient(clientConfig);

        Aws::STS::Model::AssumeRoleRequest request;
        request.SetRoleArn(role.GetArn());
        Aws::String uuid = Aws::Utils::UUID::RandomUUID();
        Aws::String roleSessionName = "iam-demo-role-session-" +

Aws::Utils::StringUtils::ToLower(uuid.c_str());
        request.SetRoleSessionName(roleSessionName);
```

```
        Aws::STS::Model::AssumeRoleOutcome assumeRoleOutcome;

        // Repeatedly call AssumeRole, because there is often a delay
        // before the role is available to be assumed.
        // Repeat at most 20 times when access is denied.
        int count = 0;
        while (true) {
            assumeRoleOutcome = stsClient.AssumeRole(request);
            if (!assumeRoleOutcome.IsSuccess()) {
                if (count > 20 ||
                    assumeRoleOutcome.GetError().GetErrorType() !=
                    Aws::STS::STSErrors::ACCESS_DENIED) {
                    std::cerr << "Error assuming role after 20 tries. " <<
                              assumeRoleOutcome.GetError().GetMessage() <<
std::endl;

                    DeleteCreatedEntities(client, role, user, policy);
                    return false;
                }
                std::this_thread::sleep_for(std::chrono::seconds(1));
            }
            else {
                std::cout << "Successfully assumed the role after " << count
                          << " seconds." << std::endl;
                break;
            }
            count++;
        }

        credentials = assumeRoleOutcome.GetResult().GetCredentials();
    }


    // 5. List objects in the bucket (This should fail).
    {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
```

```
            if (listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
std::endl;
            }
            else {
                std::cout
                        << "Access to list buckets denied because privileges have
not been applied."
                        << std::endl;
            }
        }
        else {
            std::cerr
                    << "Successfully retrieved bucket lists when this should not
happen."
                    << std::endl;
        }
    }

    // 6. Attach the policy to the role.
    {
        Aws::IAM::Model::AttachRolePolicyRequest request;
        request.SetRoleName(role.GetRoleName());
        request.WithPolicyArn(policy.GetArn());

        Aws::IAM::Model::AttachRolePolicyOutcome outcome =
client.AttachRolePolicy(
                request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error creating policy. " <<
                      outcome.GetError().GetMessage() << std::endl;

            DeleteCreatedEntities(client, role, user, policy);
            return false;
        }
        else {
            std::cout << "Successfully attached the policy with name, "
                      << policy.GetPolicyName() <<
                      ", to the role, " << role.GetRoleName() << "." <<
std::endl;
        }
    }
```

```
    int count = 0;
    // 7. List objects in the bucket (this should succeed).
    // Repeatedly call ListBuckets, because there is often a delay
    // before the policy with ListBucket permissions has been applied to the
 role.
    // Repeat at most LIST_BUCKETS_WAIT_SEC times when access is denied.
    while (true) {
        Aws::S3::S3Client s3Client(
                Aws::Auth::AWSCredentials(credentials.GetAccessKeyId(),
                                          credentials.GetSecretAccessKey(),
                                          credentials.GetSessionToken()),
                Aws::MakeShared<Aws::S3::S3EndpointProvider>(ALLOCATION_TAG),
                clientConfig);
        Aws::S3::Model::ListBucketsOutcome listBucketsOutcome =
 s3Client.ListBuckets();
        if (!listBucketsOutcome.IsSuccess()) {
            if ((count > LIST_BUCKETS_WAIT_SEC) ||
                listBucketsOutcome.GetError().GetErrorType() !=
                Aws::S3::S3Errors::ACCESS_DENIED) {
                std::cerr << "Could not lists buckets after " <<
 LIST_BUCKETS_WAIT_SEC << " seconds. " <<
                          listBucketsOutcome.GetError().GetMessage() <<
 std::endl;
                DeleteCreatedEntities(client, role, user, policy);
                return false;
            }

            std::this_thread::sleep_for(std::chrono::seconds(1));
        }
        else {

            std::cout << "Successfully retrieved bucket lists after " << count
                      << " seconds." << std::endl;
            break;
        }
        count++;
    }

    // 8. Delete all the created resources.
    return DeleteCreatedEntities(client, role, user, policy);
}

bool AwsDoc::IAM::DeleteCreatedEntities(const Aws::IAM::IAMClient &client,
```

```
                                        const Aws::IAM::Model::Role &role,
                                        const Aws::IAM::Model::User &user,
                                        const Aws::IAM::Model::Policy &policy) {
   bool result = true;
   if (policy.ArnHasBeenSet()) {
       // Detach the policy from the role.
       {
           Aws::IAM::Model::DetachRolePolicyRequest request;
           request.SetPolicyArn(policy.GetArn());
           request.SetRoleName(role.GetRoleName());

           Aws::IAM::Model::DetachRolePolicyOutcome outcome =
client.DetachRolePolicy(
                   request);
           if (!outcome.IsSuccess()) {
               std::cerr << "Error Detaching policy from roles. " <<
                         outcome.GetError().GetMessage() << std::endl;
               result = false;
           }
           else {
               std::cout << "Successfully detached the policy with arn "
                         << policy.GetArn()
                         << " from role " << role.GetRoleName() << "." <<
std::endl;
           }
       }

       // Delete the policy.
       {
           Aws::IAM::Model::DeletePolicyRequest request;
           request.WithPolicyArn(policy.GetArn());

           Aws::IAM::Model::DeletePolicyOutcome outcome =
client.DeletePolicy(request);
           if (!outcome.IsSuccess()) {
               std::cerr << "Error deleting policy. " <<
                         outcome.GetError().GetMessage() << std::endl;
               result = false;
           }
           else {
               std::cout << "Successfully deleted the policy with arn "
                         << policy.GetArn() << std::endl;
           }
       }
```

```
    }

    if (role.RoleIdHasBeenSet()) {
        // Delete the role.
        Aws::IAM::Model::DeleteRoleRequest request;
        request.SetRoleName(role.GetRoleName());

        Aws::IAM::Model::DeleteRoleOutcome outcome = client.DeleteRole(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting role. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the role with name "
                      << role.GetRoleName() << std::endl;
        }
    }

    if (user.ArnHasBeenSet()) {
        // Delete the user.
        Aws::IAM::Model::DeleteUserRequest request;
        request.WithUserName(user.GetUserName());

        Aws::IAM::Model::DeleteUserOutcome outcome = client.DeleteUser(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Error deleting user. " <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Successfully deleted the user with name "
                      << user.GetUserName() << std::endl;
        }
    }

    return result;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy

- CreateAccessKey
- CreatePolicy
- CreateRole
- CreateUser
- DeleteAccessKey
- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "math/rand"
 "strings"
 "time"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/credentials"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/aws-sdk-go-v2/service/s3"
 "github.com/aws/aws-sdk-go-v2/service/sts"
 "github.com/aws/smithy-go"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/iam/actions"
)

// AssumeRoleScenario shows you how to use the AWS Identity and Access Management
 (IAM)
// service to perform the following actions:
//
//  1. Create a user who has no permissions.
//  2. Create a role that grants permission to list Amazon Simple Storage Service
//     (Amazon S3) buckets for the account.
//  3. Add a policy to let the user assume the role.
//  4. Try and fail to list buckets without permissions.
//  5. Assume the role and list S3 buckets using temporary credentials.
//  6. Delete the policy, role, and user.
type AssumeRoleScenario struct {
 sdkConfig      aws.Config
 accountWrapper actions.AccountWrapper
 policyWrapper  actions.PolicyWrapper
 roleWrapper    actions.RoleWrapper
 userWrapper    actions.UserWrapper
 questioner     demotools.IQuestioner
 helper         IScenarioHelper
 isTestRun      bool
}

// NewAssumeRoleScenario constructs an AssumeRoleScenario instance from a
 configuration.
// It uses the specified config to get an IAM client and create wrappers for the
 actions
// used in the scenario.
func NewAssumeRoleScenario(sdkConfig aws.Config, questioner
 demotools.IQuestioner,
 helper IScenarioHelper) AssumeRoleScenario {
 iamClient := iam.NewFromConfig(sdkConfig)
 return AssumeRoleScenario{
  sdkConfig:      sdkConfig,
  accountWrapper: actions.AccountWrapper{IamClient: iamClient},
  policyWrapper:  actions.PolicyWrapper{IamClient: iamClient},
```

```
  roleWrapper:    actions.RoleWrapper{IamClient: iamClient},
  userWrapper:    actions.UserWrapper{IamClient: iamClient},
  questioner:     questioner,
  helper:         helper,
 }
}

// addTestOptions appends the API options specified in the original configuration
 to
// another configuration. This is used to attach the middleware stubber to
 clients
// that are constructed during the scenario, which is needed for unit testing.
func (scenario AssumeRoleScenario) addTestOptions(scenarioConfig *aws.Config) {
 if scenario.isTestRun {
  scenarioConfig.APIOptions = append(scenarioConfig.APIOptions,
 scenario.sdkConfig.APIOptions...)
 }
}

// Run runs the interactive scenario.
func (scenario AssumeRoleScenario) Run(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong with the demo.\n")
   log.Println(r)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Println("Welcome to the AWS Identity and Access Management (IAM) assume role
 demo.")
 log.Println(strings.Repeat("-", 88))

 user := scenario.CreateUser(ctx)
 accessKey := scenario.CreateAccessKey(ctx, user)
 role := scenario.CreateRoleAndPolicies(ctx, user)
 noPermsConfig := scenario.ListBucketsWithoutPermissions(ctx, accessKey)
 scenario.ListBucketsWithAssumedRole(ctx, noPermsConfig, role)
 scenario.Cleanup(ctx, user, role)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

```
// CreateUser creates a new IAM user. This user has no permissions.
func (scenario AssumeRoleScenario) CreateUser(ctx context.Context) *types.User {
 log.Println("Let's create an example user with no permissions.")
 userName := scenario.questioner.Ask("Enter a name for the example user:",
 demotools.NotEmpty{})
 user, err := scenario.userWrapper.GetUser(ctx, userName)
 if err != nil {
  panic(err)
 }
 if user == nil {
  user, err = scenario.userWrapper.CreateUser(ctx, userName)
  if err != nil {
   panic(err)
  }
  log.Printf("Created user %v.\n", *user.UserName)
 } else {
  log.Printf("User %v already exists.\n", *user.UserName)
 }
 log.Println(strings.Repeat("-", 88))
 return user
}

// CreateAccessKey creates an access key for the user.
func (scenario AssumeRoleScenario) CreateAccessKey(ctx context.Context, user
 *types.User) *types.AccessKey {
 accessKey, err := scenario.userWrapper.CreateAccessKeyPair(ctx, *user.UserName)
 if err != nil {
  panic(err)
 }
 log.Printf("Created access key %v for your user.", *accessKey.AccessKeyId)
 log.Println("Waiting a few seconds for your user to be ready...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return accessKey
}

// CreateRoleAndPolicies creates a policy that grants permission to list S3
 buckets for
// the current account and attaches the policy to a newly created role. It also
 adds an
// inline policy to the specified user that grants the user permission to assume
 the role.
```

```
func (scenario AssumeRoleScenario) CreateRoleAndPolicies(ctx context.Context,
 user *types.User) *types.Role {
 log.Println("Let's create a role and policy that grant permission to list S3
 buckets.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 listBucketsRole, err := scenario.roleWrapper.CreateRole(ctx,
 scenario.helper.GetName(), *user.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created role %v.\n", *listBucketsRole.RoleName)
 listBucketsPolicy, err := scenario.policyWrapper.CreatePolicy(
  ctx, scenario.helper.GetName(), []string{"s3:ListAllMyBuckets"},
 "arn:aws:s3:::*")
 if err != nil {
  panic(err)
 }
 log.Printf("Created policy %v.\n", *listBucketsPolicy.PolicyName)
 err = scenario.roleWrapper.AttachRolePolicy(ctx, *listBucketsPolicy.Arn,
 *listBucketsRole.RoleName)
 if err != nil {
  panic(err)
 }
 log.Printf("Attached policy %v to role %v.\n", *listBucketsPolicy.PolicyName,
  *listBucketsRole.RoleName)
 err = scenario.userWrapper.CreateUserPolicy(ctx, *user.UserName,
 scenario.helper.GetName(),
  []string{"sts:AssumeRole"}, *listBucketsRole.Arn)
 if err != nil {
  panic(err)
 }
 log.Printf("Created an inline policy for user %v that lets the user assume the
 role.\n",
  *user.UserName)
 log.Println("Let's give AWS a few seconds to propagate these new resources and
 connections...")
 scenario.helper.Pause(10)
 log.Println(strings.Repeat("-", 88))
 return listBucketsRole
}

// ListBucketsWithoutPermissions creates an Amazon S3 client from the user's
 access key
```

```
// credentials and tries to list buckets for the account. Because the user does
 not have
// permission to perform this action, the action fails.
func (scenario AssumeRoleScenario) ListBucketsWithoutPermissions(ctx
 context.Context, accessKey *types.AccessKey) *aws.Config {
 log.Println("Let's try to list buckets without permissions. This should return
 an AccessDenied error.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 noPermsConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *accessKey.AccessKeyId, *accessKey.SecretAccessKey, ""),
  ))
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&noPermsConfig)

 s3Client := s3.NewFromConfig(noPermsConfig)
 _, err = s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  // The SDK for Go does not model the AccessDenied error, so check ErrorCode
 directly.
  var ae smithy.APIError
  if errors.As(err, &ae) {
   switch ae.ErrorCode() {
   case "AccessDenied":
    log.Println("Got AccessDenied error, which is the expected result because\n"
 +
     "the ListBuckets call was made without permissions.")
   default:
    log.Println("Expected AccessDenied, got something else.")
    panic(err)
   }
  }
 } else {
  log.Println("Expected AccessDenied error when calling ListBuckets without
 permissions,\n" +
   "but the call succeeded. Continuing the example anyway...")
 }
 log.Println(strings.Repeat("-", 88))
 return &noPermsConfig
```

```
}

// ListBucketsWithAssumedRole performs the following actions:
//
//  1. Creates an AWS Security Token Service (AWS STS) client from the config
 created from
//     the user's access key credentials.
//  2. Gets temporary credentials by assuming the role that grants permission to
 list the
//     buckets.
//  3. Creates an Amazon S3 client from the temporary credentials.
//  4. Lists buckets for the account. Because the temporary credentials are
 generated by
//     assuming the role that grants permission, the action succeeds.
func (scenario AssumeRoleScenario) ListBucketsWithAssumedRole(ctx
 context.Context, noPermsConfig *aws.Config, role *types.Role) {
 log.Println("Let's assume the role that grants permission to list buckets and
 try again.")
 scenario.questioner.Ask("Press Enter when you're ready.")
 stsClient := sts.NewFromConfig(*noPermsConfig)
 tempCredentials, err := stsClient.AssumeRole(ctx, &sts.AssumeRoleInput{
  RoleArn:         role.Arn,
  RoleSessionName: aws.String("AssumeRoleExampleSession"),
  DurationSeconds: aws.Int32(900),
 })
 if err != nil {
  log.Printf("Couldn't assume role %v.\n", *role.RoleName)
  panic(err)
 }
 log.Printf("Assumed role %v, got temporary credentials.\n", *role.RoleName)
 assumeRoleConfig, err := config.LoadDefaultConfig(ctx,
  config.WithCredentialsProvider(credentials.NewStaticCredentialsProvider(
   *tempCredentials.Credentials.AccessKeyId,
   *tempCredentials.Credentials.SecretAccessKey,
   *tempCredentials.Credentials.SessionToken),
  ),
 )
 if err != nil {
  panic(err)
 }

 // Add test options if this is a test run. This is needed only for testing
 purposes.
 scenario.addTestOptions(&assumeRoleConfig)
```

```
 s3Client := s3.NewFromConfig(assumeRoleConfig)
 result, err := s3Client.ListBuckets(ctx, &s3.ListBucketsInput{})
 if err != nil {
  log.Println("Couldn't list buckets with assumed role credentials.")
  panic(err)
 }
 log.Println("Successfully called ListBuckets with assumed role credentials, \n"
 +
  "here are some of them:")
 for i := 0; i < len(result.Buckets) && i < 5; i++ {
  log.Printf("\t%v\n", *result.Buckets[i].Name)
 }
 log.Println(strings.Repeat("-", 88))
}

// Cleanup deletes all resources created for the scenario.
func (scenario AssumeRoleScenario) Cleanup(ctx context.Context, user *types.User,
 role *types.Role) {
 if scenario.questioner.AskBool(
  "Do you want to delete the resources created for this example? (y/n)", "y",
 ) {
  policies, err := scenario.roleWrapper.ListAttachedRolePolicies(ctx,
 *role.RoleName)
  if err != nil {
   panic(err)
  }
  for _, policy := range policies {
   err = scenario.roleWrapper.DetachRolePolicy(ctx, *role.RoleName,
 *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   err = scenario.policyWrapper.DeletePolicy(ctx, *policy.PolicyArn)
   if err != nil {
    panic(err)
   }
   log.Printf("Detached policy %v from role %v and deleted the policy.\n",
    *policy.PolicyName, *role.RoleName)
  }
  err = scenario.roleWrapper.DeleteRole(ctx, *role.RoleName)
  if err != nil {
   panic(err)
  }
```

```
  log.Printf("Deleted role %v.\n", *role.RoleName)

  userPols, err := scenario.userWrapper.ListUserPolicies(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, userPol := range userPols {
   err = scenario.userWrapper.DeleteUserPolicy(ctx, *user.UserName, userPol)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted policy %v from user %v.\n", userPol, *user.UserName)
  }
  keys, err := scenario.userWrapper.ListAccessKeys(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  for _, key := range keys {
   err = scenario.userWrapper.DeleteAccessKey(ctx, *user.UserName,
 *key.AccessKeyId)
   if err != nil {
    panic(err)
   }
   log.Printf("Deleted access key %v from user %v.\n", *key.AccessKeyId,
 *user.UserName)
  }
  err = scenario.userWrapper.DeleteUser(ctx, *user.UserName)
  if err != nil {
   panic(err)
  }
  log.Printf("Deleted user %v.\n", *user.UserName)
  log.Println(strings.Repeat("-", 88))
 }

}

// IScenarioHelper abstracts input and wait functions from a scenario so that
 they
// can be mocked for unit testing.
type IScenarioHelper interface {
 GetName() string
 Pause(secs int)
}
```

```
const rMax = 100000

type ScenarioHelper struct {
 Prefix string
 Random *rand.Rand
}

// GetName returns a unique name formed of a prefix and a random number.
func (helper *ScenarioHelper) GetName() string {
 return fmt.Sprintf("%v%v", helper.Prefix, helper.Random.Intn(rMax))
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 time.Sleep(time.Duration(secs) * time.Second)
}
```

定义一个封装账户操作的结构。

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
```

```
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}



// ListSAMLProviders gets the SAML providers for the account.
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

定义一个封装策略操作的结构。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
 }
 return policies, err
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}
```

```
// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
```

```
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

定义一个封装角色操作的结构。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}
```

```
// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
```

```
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
```

```
  role = result.Role
 }
 return role, err
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}



// AttachRolePolicy attaches a policy to a role.
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
```

```
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
  policies = result.PolicyNames
```

```
 }
 return policies, err
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

定义一个封装用户操作的结构。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}
```

```
// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
}
```

```
// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
```

```
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

```
// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
```

```
 }
 return err
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 用户操作的函数。

```
/*
  To run this Java V2 code example, set up your development environment,
 including your credentials.

  For information, see this documentation topic:

  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html

  This example performs these operations:

  1. Creates a user that has no permissions.
  2. Creates a role and policy that grants Amazon S3 permissions.
  3. Creates a role.
  4. Grants the user permissions.
  5. Gets temporary credentials by assuming the role.  Creates an Amazon S3
 Service client object with the temporary credentials.
  6. Deletes the resources.
 */

public class IAMScenario {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");
    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
            "            \"s3:*\"" +
            "       ]," +
```

```
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

   public static String userArn;

   public static void main(String[] args) throws Exception {

       final String usage = """

               Usage:
                   <username> <policyName> <roleName> <roleSessionName>
<bucketName>\s

               Where:
                   username - The name of the IAM user to create.\s
                   policyName - The name of the policy to create.\s
                   roleName - The name of the role to create.\s
                   roleSessionName - The name of the session required for the
assumeRole operation.\s
                   bucketName - The name of the Amazon S3 bucket from which
objects are read.\s
               """;

       if (args.length != 5) {
           System.out.println(usage);
           System.exit(1);
       }

       String userName = args[0];
       String policyName = args[1];
       String roleName = args[2];
       String roleSessionName = args[3];
       String bucketName = args[4];

       Region region = Region.AWS_GLOBAL;
       IamClient iam = IamClient.builder()
               .region(region)
               .build();

       System.out.println(DASHES);
       System.out.println("Welcome to the AWS IAM example scenario.");
       System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println(" 1. Create the IAM user.");
        User createUser = createIAMUser(iam, userName);

        System.out.println(DASHES);
        userArn = createUser.arn();

        AccessKey myKey = createIAMAccessKey(iam, userName);
        String accessKey = myKey.accessKeyId();
        String secretKey = myKey.secretAccessKey();
        String assumeRolePolicyDocument = "{" +
                "\"Version\": \"2012-10-17\"," +
                "\"Statement\": [{" +
                "\"Effect\": \"Allow\"," +
                "\"Principal\": {" +
                " \"AWS\": \"" + userArn + "\"" +
                "}," +
                "\"Action\": \"sts:AssumeRole\"" +
                "}]" +
                "}";

        System.out.println(assumeRolePolicyDocument);
        System.out.println(userName + " was successfully created.");
        System.out.println(DASHES);
        System.out.println("2. Creates a policy.");
        String polArn = createIAMPolicy(iam, policyName);
        System.out.println("The policy " + polArn + " was successfully
created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("3. Creates a role.");
        TimeUnit.SECONDS.sleep(30);
        String roleArn = createIAMRole(iam, roleName, assumeRolePolicyDocument);
        System.out.println(roleArn + " was successfully created.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Grants the user permissions.");
        attachIAMRolePolicy(iam, roleName, polArn);
        System.out.println(DASHES);

        System.out.println(DASHES);
```

```
        System.out.println("*** Wait for 30 secs so the resource is available");
        TimeUnit.SECONDS.sleep(30);
        System.out.println("5. Gets temporary credentials by assuming the
role.");
        System.out.println("Perform an Amazon S3 Service operation using the
temporary credentials.");
        assumeRole(roleArn, roleSessionName, bucketName, accessKey, secretKey);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6 Getting ready to delete the AWS resources");
        deleteKey(iam, userName, accessKey);
        deleteRole(iam, roleName, polArn);
        deleteIAMUser(iam, userName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("This IAM Scenario has successfully completed");
        System.out.println(DASHES);
    }

    public static AccessKey createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static User createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object
            IamWaiter iamWaiter = iam.waiter();
            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();
```

```
            // Wait until the user is created.
            CreateUserResponse response = iam.createUser(request);
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
iamWaiter.waitUntilUserExists(userRequest);

waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return null;
    }

    public static String createIAMRole(IamClient iam, String rolename, String
json) {

        try {
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(json)
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
response.role().arn());
            return response.role().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
```

```
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();
            CreatePolicyRequest request = CreatePolicyRequest.builder()
                    .policyName(policyName)
                    .policyDocument(PolicyDocument).build();

            CreatePolicyResponse response = iam.createPolicy(request);
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
iamWaiter.waitUntilPolicyExists(polRequest);

waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();

            ListAttachedRolePoliciesResponse response =
iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();
            String polArn;
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
this role.");
                    return;
                }
            }
```

```
            AttachRolePolicyRequest attachRequest =
AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);
            System.out.println("Successfully attached policy " + policyArn + " to
role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Invoke an Amazon S3 operation using the Assumed Role.
    public static void assumeRole(String roleArn, String roleSessionName, String
bucketName, String keyVal,
            String keySecret) {

        // Use the creds of the new IAM user that was created in this code
example.
        AwsBasicCredentials credentials = AwsBasicCredentials.create(keyVal,
keySecret);
        StsClient stsClient = StsClient.builder()
                .region(Region.US_EAST_1)

.credentialsProvider(StaticCredentialsProvider.create(credentials))
                .build();

        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();
            String key = myCreds.accessKeyId();
            String secKey = myCreds.secretAccessKey();
            String secToken = myCreds.sessionToken();
```

```
            // List all objects in an Amazon S3 bucket using the temp creds
retrieved by
            // invoking assumeRole.
            Region region = Region.US_EAST_1;
            S3Client s3 = S3Client.builder()
                    .credentialsProvider(

StaticCredentialsProvider.create(AwsSessionCredentials.create(key, secKey,
secToken)))
                    .region(region)
                    .build();

            System.out.println("Created a S3Client using temp credentials.");
            System.out.println("Listing objects in " + bucketName);
            ListObjectsRequest listObjects = ListObjectsRequest.builder()
                    .bucket(bucketName)
                    .build();

            ListObjectsResponse res = s3.listObjects(listObjects);
            List<S3Object> objects = res.contents();
            for (S3Object myValue : objects) {
                System.out.println("The name of the key is " + myValue.key());
                System.out.println("The owner is " + myValue.owner());
            }

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public static void deleteRole(IamClient iam, String roleName, String polArn)
{

        try {
            // First the policy needs to be detached.
            DetachRolePolicyRequest rolePolicyRequest =
DetachRolePolicyRequest.builder()
                    .policyArn(polArn)
                    .roleName(roleName)
                    .build();

            iam.detachRolePolicy(rolePolicyRequest);
```

```
            // Delete the policy.
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(polArn)
                    .build();

            iam.deletePolicy(request);
            System.out.println("*** Successfully deleted " + polArn);

            // Delete the role.
            DeleteRoleRequest roleRequest = DeleteRoleRequest.builder()
                    .roleName(roleName)
                    .build();

            iam.deleteRole(roleRequest);
            System.out.println("*** Successfully deleted " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteKey(IamClient iam, String username, String
accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
```

```
                    .build();

            iam.deleteUser(request);
            System.out.println("*** Successfully deleted " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
import {
  CreateUserCommand,
  GetUserCommand,
  CreateAccessKeyCommand,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  DeleteAccessKeyCommand,
  DeleteUserCommand,
  DeleteRoleCommand,
  DeletePolicyCommand,
  DetachRolePolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";
import { ListBucketsCommand, S3Client } from "@aws-sdk/client-s3";
import { AssumeRoleCommand, STSClient } from "@aws-sdk/client-sts";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import { ScenarioInput } from "@aws-doc-sdk-examples/lib/scenario/index.js";

// Set the parameters.
const iamClient = new IAMClient({});
const userName = "iam_basic_test_username";
const policyName = "iam_basic_test_policy";
const roleName = "iam_basic_test_role";

/**
 * Create a new IAM user. If the user already exists, give
 * the option to delete and re-create it.
 * @param {string} name
 */
export const createUser = async (name, confirmAll = false) => {
  try {
    const { User } = await iamClient.send(
      new GetUserCommand({ UserName: name }),
    );
    const input = new ScenarioInput(
      "deleteUser",
      "Do you want to delete and remake this user?",
      { type: "confirm" },
    );
```

```
    const deleteUser = await input.handle({}, { confirmAll });
    // If the user exists, and you want to delete it, delete the user
    // and then create it again.
    if (deleteUser) {
      await iamClient.send(new DeleteUserCommand({ UserName: User.UserName }));
      await iamClient.send(new CreateUserCommand({ UserName: name }));
    } else {
      console.warn(
        `${name} already exists. The scenario may not work as expected.`,
      );
      return User;
    }
  } catch (caught) {
    // If there is no user by that name, create one.
    if (caught instanceof Error && caught.name === "NoSuchEntityException") {
      const { User } = await iamClient.send(
        new CreateUserCommand({ UserName: name }),
      );
      return User;
    }
    throw caught;
  }
};

export const main = async (confirmAll = false) => {
  // Create a user. The user has no permissions by default.
  const User = await createUser(userName, confirmAll);

  if (!User) {
    throw new Error("User not created");
  }

  // Create an access key. This key is used to authenticate the new user to
  // Amazon Simple Storage Service (Amazon S3) and AWS Security Token Service
 (AWS STS).
  // It's not best practice to use access keys. For more information, see
 https://aws.amazon.com/iam/resources/best-practices/.
  const createAccessKeyResponse = await iamClient.send(
    new CreateAccessKeyCommand({ UserName: userName }),
  );

  if (
    !createAccessKeyResponse.AccessKey?.AccessKeyId ||
    !createAccessKeyResponse.AccessKey?.SecretAccessKey
```

```
  ) {
    throw new Error("Access key not created");
  }

  const {
    AccessKey: { AccessKeyId, SecretAccessKey },
  } = createAccessKeyResponse;

  let s3Client = new S3Client({
    credentials: {
      accessKeyId: AccessKeyId,
      secretAccessKey: SecretAccessKey,
    },
  });

  // Retry the list buckets operation until it succeeds. InvalidAccessKeyId is
  // thrown while the user and access keys are still stabilizing.
  await retry({ intervalInMs: 1000, maxRetries: 300 }, async () => {
    try {
      return await listBuckets(s3Client);
    } catch (err) {
      if (err instanceof Error && err.name === "InvalidAccessKeyId") {
        throw err;
      }
    }
  });

  // Retry the create role operation until it succeeds. A MalformedPolicyDocument
error
  // is thrown while the user and access keys are still stabilizing.
  const { Role } = await retry(
    {
      intervalInMs: 2000,
      maxRetries: 60,
    },
    () =>
      iamClient.send(
        new CreateRoleCommand({
          AssumeRolePolicyDocument: JSON.stringify({
            Version: "2012-10-17",
            Statement: [
              {
                Effect: "Allow",
                Principal: {
```

```
                // Allow the previously created user to assume this role.
                AWS: User.Arn,
              },
              Action: "sts:AssumeRole",
            },
          ],
        }),
        RoleName: roleName,
      }),
    ),
);

if (!Role) {
  throw new Error("Role not created");
}

// Create a policy that allows the user to list S3 buckets.
const { Policy: listBucketPolicy } = await iamClient.send(
  new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: ["s3:ListAllMyBuckets"],
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  }),
);

if (!listBucketPolicy) {
  throw new Error("Policy not created");
}

// Attach the policy granting the 's3:ListAllMyBuckets' action to the role.
await iamClient.send(
  new AttachRolePolicyCommand({
    PolicyArn: listBucketPolicy.Arn,
    RoleName: Role.RoleName,
  }),
);
```

```
// Assume the role.
const stsClient = new STSClient({
  credentials: {
    accessKeyId: AccessKeyId,
    secretAccessKey: SecretAccessKey,
  },
});

// Retry the assume role operation until it succeeds.
const { Credentials } = await retry(
  { intervalInMs: 2000, maxRetries: 60 },
  () =>
    stsClient.send(
      new AssumeRoleCommand({
        RoleArn: Role.Arn,
        RoleSessionName: `iamBasicScenarioSession-${Math.floor(
          Math.random() * 1000000,
        )}`,
        DurationSeconds: 900,
      }),
    ),
);

if (!Credentials?.AccessKeyId || !Credentials?.SecretAccessKey) {
  throw new Error("Credentials not created");
}

s3Client = new S3Client({
  credentials: {
    accessKeyId: Credentials.AccessKeyId,
    secretAccessKey: Credentials.SecretAccessKey,
    sessionToken: Credentials.SessionToken,
  },
});

// List the S3 buckets again.
// Retry the list buckets operation until it succeeds. AccessDenied might
// be thrown while the role policy is still stabilizing.
await retry({ intervalInMs: 2000, maxRetries: 120 }, () =>
  listBuckets(s3Client),
);

// Clean up.
```

```
  await iamClient.send(
    new DetachRolePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeletePolicyCommand({
      PolicyArn: listBucketPolicy.Arn,
    }),
  );

  await iamClient.send(
    new DeleteRoleCommand({
      RoleName: Role.RoleName,
    }),
  );

  await iamClient.send(
    new DeleteAccessKeyCommand({
      UserName: userName,
      AccessKeyId,
    }),
  );

  await iamClient.send(
    new DeleteUserCommand({
      UserName: userName,
    }),
  );
};

/**
 *
 * @param {S3Client} s3Client
 */
const listBuckets = async (s3Client) => {
  const { Buckets } = await s3Client.send(new ListBucketsCommand({}));

  if (!Buckets) {
    throw new Error("Buckets not listed");
  }
```

```
  console.log(Buckets.map((bucket) => bucket.Name).join("\n"));
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

创建包装 IAM 用户操作的函数。

```
suspend fun main(args: Array<String>) {
    val usage = """
    Usage:
```

```
        <username> <policyName> <roleName> <roleSessionName> <fileLocation>
<bucketName>

   Where:
        username - The name of the IAM user to create.
        policyName - The name of the policy to create.
        roleName - The name of the role to create.
        roleSessionName - The name of the session required for the assumeRole
operation.
        fileLocation - The file location to the JSON required to create the role
(see Readme).
        bucketName - The name of the Amazon S3 bucket from which objects are
read.
   """

   if (args.size != 6) {
        println(usage)
        exitProcess(1)
   }

   val userName = args[0]
   val policyName = args[1]
   val roleName = args[2]
   val roleSessionName = args[3]
   val fileLocation = args[4]
   val bucketName = args[5]

   createUser(userName)
   println("$userName was successfully created.")

   val polArn = createPolicy(policyName)
   println("The policy $polArn was successfully created.")

   val roleArn = createRole(roleName, fileLocation)
   println("$roleArn was successfully created.")
   attachRolePolicy(roleName, polArn)

   println("*** Wait for 1 MIN so the resource is available.")
   delay(60000)
   assumeGivenRole(roleArn, roleSessionName, bucketName)

   println("*** Getting ready to delete the AWS resources.")
   deleteRole(roleName, polArn)
   deleteUser(userName)
```

```
    println("This IAM Scenario has successfully completed.")
}

suspend fun createUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}

suspend fun createPolicy(policyNameVal: String?): String {
    val policyDocumentValue = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "s3:*"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentValue
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}

suspend fun createRole(
```

```
    rolenameVal: String?,
    fileLocation: String?,
): String? {
    val jsonObject = fileLocation?.let { readJsonSimpleDemo(it) } as JSONObject

    val request =
        CreateRoleRequest {
            roleName = rolenameVal
            assumeRolePolicyDocument = jsonObject.toJSONString()
            description = "Created using the AWS SDK for Kotlin"
        }

    IamClient { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createRole(request)
        return response.role?.arn
    }
}

suspend fun attachRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkMyList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
```

```
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkMyList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}

suspend fun assumeGivenRole(
    roleArnVal: String?,
    roleSessionNameVal: String?,
    bucketName: String,
) {
    val stsClient = StsClient.fromEnvironment { region = "us-east-1" }
    val roleRequest =
        AssumeRoleRequest {
            roleArn = roleArnVal
            roleSessionName = roleSessionNameVal
        }

    val roleResponse = stsClient.assumeRole(roleRequest)
    val myCreds = roleResponse.credentials
    val key = myCreds?.accessKeyId
    val secKey = myCreds?.secretAccessKey
    val secToken = myCreds?.sessionToken

    val staticCredentials = StaticCredentialsProvider {
        accessKeyId = key
        secretAccessKey = secKey
        sessionToken = secToken
```

```
    }

    // List all objects in an Amazon S3 bucket using the temp creds.
    val s3 = S3Client.fromEnvironment {
        region = "us-east-1"
        credentialsProvider = staticCredentials
    }

    println("Created a S3Client using temp credentials.")
    println("Listing objects in $bucketName")

    val listObjects =
        ListObjectsRequest {
            bucket = bucketName
        }

    val response = s3.listObjects(listObjects)
    response.contents?.forEach { myObject ->
        println("The name of the key is ${myObject.key}")
        println("The owner is ${myObject.owner}")
    }
}

suspend fun deleteRole(
    roleNameVal: String,
    polArn: String,
) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }

    // First the policy needs to be detached.
    val rolePolicyRequest =
        DetachRolePolicyRequest {
            policyArn = polArn
            roleName = roleNameVal
        }

    iam.detachRolePolicy(rolePolicyRequest)

    // Delete the policy.
    val request =
        DeletePolicyRequest {
            policyArn = polArn
        }
```

```
    iam.deletePolicy(request)
    println("*** Successfully deleted $polArn")

    // Delete the role.
    val roleRequest =
        DeleteRoleRequest {
            roleName = roleNameVal
        }

    iam.deleteRole(roleRequest)
    println("*** Successfully deleted $roleNameVal")
}

suspend fun deleteUser(userNameVal: String) {
    val iam = IamClient.fromEnvironment { region = "AWS_GLOBAL" }
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    iam.deleteUser(request)
    println("*** Successfully deleted $userNameVal")
}

@Throws(java.lang.Exception::class)
fun readJsonSimpleDemo(filename: String): Any? {
    val reader = FileReader(filename)
    val jsonParser = JSONParser()
    return jsonParser.parse(reader)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy

- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
namespace Iam\Basics;

require 'vendor/autoload.php';

use Aws\Credentials\Credentials;
use Aws\S3\Exception\S3Exception;
use Aws\S3\S3Client;
use Aws\Sts\StsClient;
use Iam\IAMService;

echo("\n");
echo("---------------------------------------\n");
print("Welcome to the IAM getting started demo using PHP!\n");
echo("---------------------------------------\n");

$uuid = uniqid();
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";

$key = $service->createAccessKey($user['UserName']);
$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
```

```
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

$inlinePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{$assumeRoleRole['Arn']}\"}]
}";
$inlinePolicy = $service->createUserPolicy("iam_demo_inline_policy_$uuid",
 $inlinePolicyDocument, $user['UserName']);
//First, fail to list the buckets with the user
$credentials = new Credentials($key['AccessKeyId'], $key['SecretAccessKey']);
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
try {
    $s3Client->listBuckets([
    ]);
    echo "this should not run";
} catch (S3Exception $exception) {
    echo "successfully failed!\n";
}
```

```
$stsClient = new StsClient(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $credentials]);
sleep(10);
$assumedRole = $stsClient->assumeRole([
    'RoleArn' => $assumeRoleRole['Arn'],
    'RoleSessionName' => "DemoAssumeRoleSession_$uuid",
]);
$assumedCredentials = [
    'key' => $assumedRole['Credentials']['AccessKeyId'],
    'secret' => $assumedRole['Credentials']['SecretAccessKey'],
    'token' => $assumedRole['Credentials']['SessionToken'],
];
$s3Client = new S3Client(['region' => 'us-west-2', 'version' => 'latest',
 'credentials' => $assumedCredentials]);
try {
    $s3Client->listBuckets([]);
    echo "this should now run!\n";
} catch (S3Exception $exception) {
    echo "this should now not fail\n";
}

$service->detachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);
$deletePolicy = $service->deletePolicy($listAllBucketsPolicy['Arn']);
echo "Delete policy: {$listAllBucketsPolicy['PolicyName']}\n";
$deletedRole = $service->deleteRole($assumeRoleRole['Arn']);
echo "Deleted role: {$assumeRoleRole['RoleName']}\n";
$deletedKey = $service->deleteAccessKey($key['AccessKeyId'], $user['UserName']);
$deletedUser = $service->deleteUser($user['UserName']);
echo "Delete user: {$user['UserName']}\n";
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey

- DeletePolicy
- DeleteRole
- DeleteUser
- DeleteUserPolicy
- DetachRolePolicy
- PutUserPolicy

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
import json
import sys
import time
from uuid import uuid4

import boto3
from botocore.exceptions import ClientError


def progress_bar(seconds):
    """Shows a simple progress bar in the command window."""
    for _ in range(seconds):
        time.sleep(1)
        print(".", end="")
        sys.stdout.flush()
    print()


def setup(iam_resource):
```

```
    """
    Creates a new user with no permissions.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role.
    Creates a policy that allows listing Amazon S3 buckets.
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, and role.
    """
    try:
        user = iam_resource.create_user(UserName=f"demo-user-{uuid4()}")
        print(f"Created user {user.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a user for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        user_key = user.create_access_key_pair()
        print(f"Created access key pair for user.")
    except ClientError as error:
        print(
            f"Couldn't create access keys for user {user.name}. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    try:
        role = iam_resource.create_role(
            RoleName=f"demo-role-{uuid4()}",
            AssumeRolePolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
```

```
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Principal": {"AWS": user.arn},
                            "Action": "sts:AssumeRole",
                        }
                    ],
                }
            ),
        )
        print(f"Created role {role.name}.")
    except ClientError as error:
        print(
            f"Couldn't create a role for the demo. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        policy = iam_resource.create_policy(
            PolicyName=f"demo-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "s3:ListAllMyBuckets",
                            "Resource": "arn:aws:s3:::*",
                        }
                    ],
                }
            ),
        )
        role.attach_policy(PolicyArn=policy.arn)
        print(f"Created policy {policy.policy_name} and attached it to the
role.")
    except ClientError as error:
        print(
            f"Couldn't create a policy and attach it to role {role.name}. Here's
why: "
            f"{error.response['Error']['Message']}"
        )
        raise
```

```
    try:
        user.create_policy(
            PolicyName=f"demo-user-policy-{uuid4()}",
            PolicyDocument=json.dumps(
                {
                    "Version":"2012-10-17",
                    "Statement": [
                        {
                            "Effect": "Allow",
                            "Action": "sts:AssumeRole",
                            "Resource": role.arn,
                        }
                    ],
                }
            ),
        )
        print(
            f"Created an inline policy for {user.name} that lets the user assume
 "
            f"the role."
        )
    except ClientError as error:
        print(
            f"Couldn't create an inline policy for user {user.name}. Here's why:
 "
            f"{error.response['Error']['Message']}"
        )
        raise

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, role


def show_access_denied_without_role(user_key):
    """
    Shows that listing buckets without first assuming the role is not allowed.

    :param user_key: The key of the user created during setup. This user does not
                     have permission to list buckets in the account.
    """
```

```
    print(f"Try to list buckets without first assuming the role.")
    s3_denied_resource = boto3.resource(
        "s3", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        for bucket in s3_denied_resource.buckets.all():
            print(bucket.name)
        raise RuntimeError("Expected to get AccessDenied error when listing
 buckets!")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Attempt to list buckets with no permissions: AccessDenied.")
        else:
            raise


def list_buckets_from_assumed_role(user_key, assume_role_arn, session_name):
    """
    Assumes a role that grants permission to list the Amazon S3 buckets in the
 account.
    Uses the temporary credentials from the role to list the buckets that are
 owned
    by the assumed role's account.

    :param user_key: The access key of a user that has permission to assume the
 role.
    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    """
    sts_client = boto3.client(
        "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
    )
    try:
        response = sts_client.assume_role(
            RoleArn=assume_role_arn, RoleSessionName=session_name
        )
        temp_credentials = response["Credentials"]
        print(f"Assumed role {assume_role_arn} and got temporary credentials.")
    except ClientError as error:
        print(
            f"Couldn't assume role {assume_role_arn}. Here's why: "
```

```
            f"{error.response['Error']['Message']}"
        )
        raise

    # Create an S3 resource that can access the account with the temporary
 credentials.
    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )
    print(f"Listing buckets for the assumed role's account:")
    try:
        for bucket in s3_resource.buckets.all():
            print(bucket.name)
    except ClientError as error:
        print(
            f"Couldn't list buckets for the account. Here's why: "
            f"{error.response['Error']['Message']}"
        )
        raise




def teardown(user, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    try:
        for attached in role.attached_policies.all():
            policy_name = attached.policy_name
            role.detach_policy(PolicyArn=attached.arn)
            attached.delete()
            print(f"Detached and deleted {policy_name}.")
        role.delete()
        print(f"Deleted {role.name}.")
    except ClientError as error:
        print(
            "Couldn't detach policy, delete policy, or delete role. Here's why: "
```

```
            f"{error.response['Error']['Message']}"
        )
        raise

    try:
        for user_pol in user.policies.all():
            user_pol.delete()
            print("Deleted inline user policy.")
        for key in user.access_keys.all():
            key.delete()
            print("Deleted user's access key.")
        user.delete()
        print(f"Deleted {user.name}.")
    except ClientError as error:
        print(
            "Couldn't delete user policy or delete user. Here's why: "
            f"{error.response['Error']['Message']}"
        )


def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the IAM create user and assume role demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user = None
    role = None
    try:
        user, user_key, role = setup(iam_resource)
        print(f"Created {user.name} and {role.name}.")
        show_access_denied_without_role(user_key)
        list_buckets_from_assumed_role(user_key, role.arn,
 "AssumeRoleDemoSession")
    except Exception:
        print("Something went wrong!")
    finally:
        if user is not None and role is not None:
            teardown(user, role)
        print("Thanks for watching!")


if __name__ == "__main__":
    usage_demo()
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 IAM 用户和授予列出 Amazon S3 存储桶的权限的角色。用户仅具有代入该角色的权限。代入该角色后，使用临时凭证列出该账户的存储桶。

```
# Wraps the scenario actions.
class ScenarioCreateUserAssumeRole
  attr_reader :iam_client

  # @param [Aws::IAM::Client] iam_client: The AWS IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
```

```
    @iam_client = iam_client
    @logger = logger
  end

  # Waits for the specified number of seconds.
  #
  # @param duration [Integer] The number of seconds to wait.
  def wait(duration)
    puts('Give AWS time to propagate resources...')
    sleep(duration)
  end

  # Creates a user.
  #
  # @param user_name [String] The name to give the user.
  # @return [Aws::IAM::User] The newly created user.
  def create_user(user_name)
    user = @iam_client.create_user(user_name: user_name).user
    @logger.info("Created demo user named #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info('Tried and failed to create demo user.')
    @logger.info("\t#{e.code}: #{e.message}")
    @logger.info("\nCan't continue the demo without a user!")
    raise
  else
    user
  end

  # Creates an access key for a user.
  #
  # @param user [Aws::IAM::User] The user that owns the key.
  # @return [Aws::IAM::AccessKeyPair] The newly created access key.
  def create_access_key_pair(user)
    user_key = @iam_client.create_access_key(user_name:
user.user_name).access_key
    @logger.info("Created accesskey pair for user #{user.user_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create access keys for user #{user.user_name}.")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  else
    user_key
  end
```

```
 # Creates a role that can be assumed by a user.
 #
 # @param role_name [String] The name to give the role.
 # @param user [Aws::IAM::User] The user who is granted permission to assume the
role.
 # @return [Aws::IAM::Role] The newly created role.
 def create_role(role_name, user)
   trust_policy = {
     Version: '2012-10-17',
     Statement: [{
       Effect: 'Allow',
       Principal: { 'AWS': user.arn },
       Action: 'sts:AssumeRole'
     }]
   }.to_json
   role = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: trust_policy
   ).role
   @logger.info("Created role #{role.role_name}.")
 rescue Aws::Errors::ServiceError => e
   @logger.info("Couldn't create a role for the demo. Here's why: ")
   @logger.info("\t#{e.code}: #{e.message}")
   raise
 else
   role
 end

 # Creates a policy that grants permission to list S3 buckets in the account,
and
 # then attaches the policy to a role.
 #
 # @param policy_name [String] The name to give the policy.
 # @param role [Aws::IAM::Role] The role that the policy is attached to.
 # @return [Aws::IAM::Policy] The newly created policy.
 def create_and_attach_role_policy(policy_name, role)
   policy_document = {
     Version: '2012-10-17',
     Statement: [{
       Effect: 'Allow',
       Action: 's3:ListAllMyBuckets',
       Resource: 'arn:aws:s3:::*'
     }]
   }.to_json
```

```
    policy = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document
    ).policy
    @iam_client.attach_role_policy(
      role_name: role.role_name,
      policy_arn: policy.arn
    )
    @logger.info("Created policy #{policy.policy_name} and attached it to role
#{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create a policy and attach it to role
#{role.role_name}. Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Creates an inline policy for a user that lets the user assume a role.
  #
  # @param policy_name [String] The name to give the policy.
  # @param user [Aws::IAM::User] The user that owns the policy.
  # @param role [Aws::IAM::Role] The role that can be assumed.
  # @return [Aws::IAM::UserPolicy] The newly created policy.
  def create_user_policy(policy_name, user, role)
    policy_document = {
      Version: '2012-10-17',
      Statement: [{
        Effect: 'Allow',
        Action: 'sts:AssumeRole',
        Resource: role.arn
      }]
    }.to_json
    @iam_client.put_user_policy(
      user_name: user.user_name,
      policy_name: policy_name,
      policy_document: policy_document
    )
    puts("Created an inline policy for #{user.user_name} that lets the user
assume role #{role.role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't create an inline policy for user #{user.user_name}.
Here's why: ")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
```

```
 end

 # Creates an Amazon S3 resource with specified credentials. This is separated
into a
 # factory function so that it can be mocked for unit testing.
 #
 # @param credentials [Aws::Credentials] The credentials used by the Amazon S3
resource.
 def create_s3_resource(credentials)
   Aws::S3::Resource.new(client: Aws::S3::Client.new(credentials: credentials))
 end

 # Lists the S3 buckets for the account, using the specified Amazon S3 resource.
 # Because the resource uses credentials with limited access, it may not be able
to
 # list the S3 buckets.
 #
 # @param s3_resource [Aws::S3::Resource] An Amazon S3 resource.
 def list_buckets(s3_resource)
   count = 10
   s3_resource.buckets.each do |bucket|
     @logger.info "\t#{bucket.name}"
     count -= 1
     break if count.zero?
   end
 rescue Aws::Errors::ServiceError => e
   if e.code == 'AccessDenied'
     puts('Attempt to list buckets with no permissions: AccessDenied.')
   else
     @logger.info("Couldn't list buckets for the account. Here's why: ")
     @logger.info("\t#{e.code}: #{e.message}")
     raise
   end
 end

 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
```

```
    Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
  end

  # Gets temporary credentials that can be used to assume a role.
  #
  # @param role_arn [String] The ARN of the role that is assumed when these
credentials
  #                          are used.
  # @param sts_client [AWS::STS::Client] An AWS STS client.
  # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
  def assume_role(role_arn, sts_client)
    credentials = Aws::AssumeRoleCredentials.new(
      client: sts_client,
      role_arn: role_arn,
      role_session_name: 'create-use-assume-role-scenario'
    )
    @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
    credentials
  end

  # Deletes a role. If the role has policies attached, they are detached and
  # deleted before the role is deleted.
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    @iam_client.list_attached_role_policies(role_name:
role_name).attached_policies.each do |policy|
      @iam_client.detach_role_policy(role_name: role_name, policy_arn:
policy.policy_arn)
      @iam_client.delete_policy(policy_arn: policy.policy_arn)
      @logger.info("Detached and deleted policy #{policy.policy_name}.")
    end
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Role deleted: #{role_name}.")
  rescue Aws::Errors::ServiceError => e
    @logger.info("Couldn't detach policies and delete role #{role.name}. Here's
why:")
    @logger.info("\t#{e.code}: #{e.message}")
    raise
  end

  # Deletes a user. If the user has inline policies or access keys, they are
deleted
```

```
  # before the user is deleted.
  #
  # @param user [Aws::IAM::User] The user to delete.
  def delete_user(user_name)
    user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
    user.each do |key|
      @iam_client.delete_access_key({ access_key_id: key.access_key_id,
 user_name: user_name })
      @logger.info("Deleted access key #{key.access_key_id} for user
 '#{user_name}'.")
    end

    @iam_client.delete_user(user_name: user_name)
    @logger.info("Deleted user '#{user_name}'.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
end

# Runs the IAM create a user and assume a role scenario.
def run_scenario(scenario)
  puts('-' * 88)
  puts('Welcome to the IAM create a user and assume a role demo!')
  puts('-' * 88)
  user = scenario.create_user("doc-example-user-#{Random.uuid}")
  user_key = scenario.create_access_key_pair(user)
  scenario.wait(10)
  role = scenario.create_role("doc-example-role-#{Random.uuid}", user)
  scenario.create_and_attach_role_policy("doc-example-role-policy-
#{Random.uuid}", role)
  scenario.create_user_policy("doc-example-user-policy-#{Random.uuid}", user,
 role)
  scenario.wait(10)
  puts('Try to list buckets with credentials for a user who has no permissions.')
  puts('Expect AccessDenied from this call.')
  scenario.list_buckets(
    scenario.create_s3_resource(Aws::Credentials.new(user_key.access_key_id,
 user_key.secret_access_key))
  )
  puts('Now, assume the role that grants permission.')
  temp_credentials = scenario.assume_role(
    role.arn, scenario.create_sts_client(user_key.access_key_id,
 user_key.secret_access_key)
  )
```

```
  puts('Here are your buckets:')
  scenario.list_buckets(scenario.create_s3_resource(temp_credentials))
  puts("Deleting role '#{role.role_name}' and attached policies.")
  scenario.delete_role(role.role_name)
  puts("Deleting user '#{user.user_name}', policies, and keys.")
  scenario.delete_user(user.user_name)
  puts('Thanks for watching!')
  puts('-' * 88)
rescue Aws::Errors::ServiceError => e
  puts('Something went wrong with the demo.')
  puts("\t#{e.code}: #{e.message}")
end

run_scenario(ScenarioCreateUserAssumeRole.new(Aws::IAM::Client.new)) if
 $PROGRAM_NAME == __FILE__
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
use aws_config::meta::region::RegionProviderChain;
use aws_sdk_iam::Error as iamError;
use aws_sdk_iam::{config::Credentials as iamCredentials, config::Region, Client
 as iamClient};
use aws_sdk_s3::Client as s3Client;
use aws_sdk_sts::Client as stsClient;
use tokio::time::{sleep, Duration};
use uuid::Uuid;

#[tokio::main]
async fn main() -> Result<(), iamError> {
    let (client, uuid, list_all_buckets_policy_document, inline_policy_document)
 =
        initialize_variables().await;

    if let Err(e) = run_iam_operations(
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
    .await
    {
        println!("{:?}", e);
    };

    Ok(())
}

async fn initialize_variables() -> (iamClient, String, String, String) {
```

```
    let region_provider = RegionProviderChain::first_try(Region::new("us-
west-2"));

    let shared_config =
 aws_config::from_env().region(region_provider).load().await;
    let client = iamClient::new(&shared_config);
    let uuid = Uuid::new_v4().to_string();

    let list_all_buckets_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
    }"
    .to_string();
    let inline_policy_document = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"sts:AssumeRole\",
                    \"Resource\": \"{}\"}]
    }"
    .to_string();

    (
        client,
        uuid,
        list_all_buckets_policy_document,
        inline_policy_document,
    )
}

async fn run_iam_operations(
    client: iamClient,
    uuid: String,
    list_all_buckets_policy_document: String,
    inline_policy_document: String,
) -> Result<(), iamError> {
    let user = iam_service::create_user(&client, &format!("{}{}",
 "iam_demo_user_", uuid)).await?;
    println!("Created the user with the name: {}", user.user_name());
    let key = iam_service::create_access_key(&client, user.user_name()).await?;
```

```
    let assume_role_policy_document = "{
        \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }"
    .to_string()
    .replace("{}", user.arn());

    let assume_role_role = iam_service::create_role(
        &client,
        &format!("{}{}", "iam_demo_role_", uuid),
        &assume_role_policy_document,
    )
    .await?;
    println!("Created the role with the ARN: {}", assume_role_role.arn());

    let list_all_buckets_policy = iam_service::create_policy(
        &client,
        &format!("{}{}", "iam_demo_policy_", uuid),
        &list_all_buckets_policy_document,
    )
    .await?;
    println!(
        "Created policy: {}",
        list_all_buckets_policy.policy_name.as_ref().unwrap()
    );

    let attach_role_policy_result =
        iam_service::attach_role_policy(&client, &assume_role_role,
&list_all_buckets_policy)
            .await?;
    println!(
        "Attached the policy to the role: {:?}",
        attach_role_policy_result
    );

    let inline_policy_name = format!("{}{}", "iam_demo_inline_policy_", uuid);
    let inline_policy_document = inline_policy_document.replace("{}",
assume_role_role.arn());
    iam_service::create_user_policy(&client, &user, &inline_policy_name,
&inline_policy_document)
```

```
        .await?;
    println!("Created inline policy.");

    //First, fail to list the buckets with the user.
    let creds = iamCredentials::from_keys(key.access_key_id(),
key.secret_access_key(), None);
    let fail_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    println!("Fail config: {:?}", fail_config);
    let fail_client: s3Client = s3Client::new(&fail_config);
    match fail_client.list_buckets().send().await {
        Ok(e) => {
            println!("This should not run. {:?}", e);
        }
        Err(e) => {
            println!("Successfully failed with error: {:?}", e)
        }
    }

    let sts_config = aws_config::from_env()
        .credentials_provider(creds.clone())
        .load()
        .await;
    let sts_client: stsClient = stsClient::new(&sts_config);
    sleep(Duration::from_secs(10)).await;
    let assumed_role = sts_client
        .assume_role()
        .role_arn(assume_role_role.arn())
        .role_session_name(format!("iam_demo_assumerole_session_{uuid}"))
        .send()
        .await;
    println!("Assumed role: {:?}", assumed_role);
    sleep(Duration::from_secs(10)).await;

    let assumed_credentials = iamCredentials::from_keys(
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .access_key_id(),
```

```
        assumed_role
            .as_ref()
            .unwrap()
            .credentials
            .as_ref()
            .unwrap()
            .secret_access_key(),
        Some(
            assumed_role
                .as_ref()
                .unwrap()
                .credentials
                .as_ref()
                .unwrap()
                .session_token
                .clone(),
        ),
    );

    let succeed_config = aws_config::from_env()
        .credentials_provider(assumed_credentials)
        .load()
        .await;
    println!("succeed config: {:?}", succeed_config);
    let succeed_client: s3Client = s3Client::new(&succeed_config);
    sleep(Duration::from_secs(10)).await;
    match succeed_client.list_buckets().send().await {
        Ok(_) => {
            println!("This should now run successfully.")
        }
        Err(e) => {
            println!("This should not run. {:?}", e);
            panic!()
        }
    }

    //Clean up.
    iam_service::detach_role_policy(
        &client,
        assume_role_role.role_name(),
        list_all_buckets_policy.arn().unwrap_or_default(),
    )
    .await?;
    iam_service::delete_policy(&client, list_all_buckets_policy).await?;
```

```
    iam_service::delete_role(&client, &assume_role_role).await?;
    println!("Deleted role {}", assume_role_role.role_name());
    iam_service::delete_access_key(&client, &user, &key).await?;
    println!("Deleted key for {}", key.user_name());
    iam_service::delete_user_policy(&client, &user, &inline_policy_name).await?;
    println!("Deleted inline user policy: {}", inline_policy_name);
    iam_service::delete_user(&client, &user).await?;
    println!("Deleted user {}", user.user_name());

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateRole
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteRole
  - DeleteUser
  - DeleteUserPolicy
  - DetachRolePolicy
  - PutUserPolicy

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包的 IAM 操作

以下代码示例演示了如何使用 Amazon 开发工具包来执行各个 IAM 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 IAM API，是必须在上下文中运行的大型程序的代码节选。您可以在使用 Amazon 开发工具包的 IAM 场景中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Identity and Access Management API 参考。

示例

- 将 AddClientIdToOpenIdConnectProvider 与 CLI 配合使用
- 将 AddRoleToInstanceProfile 与 CLI 配合使用
- 将 AddUserToGroup 与 CLI 配合使用
- 将 AttachGroupPolicy 与 CLI 配合使用
- 将 AttachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 AttachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 ChangePassword 与 CLI 配合使用
- 将 CreateAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 CreateAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 CreateGroup 与 Amazon SDK 或 CLI 配合使用
- 将 CreateInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 CreateLoginProfile 与 CLI 配合使用
- 将 CreateOpenIdConnectProvider 与 CLI 配合使用
- 将 CreatePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 CreatePolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 CreateRole 与 Amazon SDK 或 CLI 配合使用
- 将 CreateSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 CreateServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 CreateUser 与 Amazon SDK 或 CLI 配合使用
- 将 CreateVirtualMfaDevice 与 CLI 配合使用
- 将 DeactivateMfaDevice 与 CLI 配合使用
- 将 DeleteAccessKey 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountAlias 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteAccountPasswordPolicy 与 CLI 配合使用
- 将 DeleteGroup 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteGroupPolicy 与 CLI 配合使用

- 将 DeleteInstanceProfile 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteLoginProfile 与 CLI 配合使用
- 将 DeleteOpenIdConnectProvider 与 CLI 配合使用
- 将 DeletePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeletePolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteRolePermissionsBoundary 与 CLI 配合使用
- 将 DeleteRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSAMLProvider 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteServiceLinkedRole 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteSigningCertificate 与 CLI 配合使用
- 将 DeleteUser 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteUserPermissionsBoundary 与 CLI 配合使用
- 将 DeleteUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DeleteVirtualMfaDevice 与 CLI 配合使用
- 将 DetachGroupPolicy 与 CLI 配合使用
- 将 DetachRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 DetachUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 EnableMfaDevice 与 CLI 配合使用
- 将 GenerateCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GenerateServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetAccessKeyLastUsed 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountAuthorizationDetails 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountPasswordPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetAccountSummary 与 Amazon SDK 或 CLI 配合使用
- 将 GetContextKeysForCustomPolicy 与 CLI 配合使用
- 将 GetContextKeysForPrincipalPolicy 与 CLI 配合使用

- 将 GetCredentialReport 与 Amazon SDK 或 CLI 配合使用
- 将 GetGroup 与 CLI 配合使用
- 将 GetGroupPolicy 与 CLI 配合使用
- 将 GetInstanceProfile 与 CLI 配合使用
- 将 GetLoginProfile 与 CLI 配合使用
- 将 GetOpenIdConnectProvider 与 CLI 配合使用
- 将 GetPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 GetPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 GetRole 与 Amazon SDK 或 CLI 配合使用
- 将 GetRolePolicy 与 CLI 配合使用
- 将 GetSamlProvider 与 CLI 配合使用
- 将 GetServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 GetServiceLastAccessedDetails 与 CLI 配合使用
- 将 GetServiceLastAccessedDetailsWithEntities 与 CLI 配合使用
- 将 GetServiceLinkedRoleDeletionStatus 与 Amazon SDK 或 CLI 配合使用
- 将 GetUser 与 Amazon SDK 或 CLI 配合使用
- 将 GetUserPolicy 与 CLI 配合使用
- 将 ListAccessKeys 与 Amazon SDK 或 CLI 配合使用
- 将 ListAccountAliases 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedGroupPolicies 与 CLI 配合使用
- 将 ListAttachedRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListAttachedUserPolicies 与 CLI 配合使用
- 将 ListEntitiesForPolicy 与 CLI 配合使用
- 将 ListGroupPolicies 与 CLI 配合使用
- 将 ListGroups 与 Amazon SDK 或 CLI 配合使用
- 将 ListGroupsForUser 与 CLI 配合使用
- 将 ListInstanceProfiles 与 CLI 配合使用
- 将 ListInstanceProfilesForRole 与 CLI 配合使用

- 将 ListMfaDevices 与 CLI 配合使用
- 将 ListOpenIdConnectProviders 与 CLI 配合使用
- 将 ListPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListPolicyVersions 与 Amazon SDK 或 CLI 配合使用
- 将 ListRolePolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListRoleTags 与 CLI 配合使用
- 将 ListRoles 与 Amazon SDK 或 CLI 配合使用
- 将 ListSAMLProviders 与 Amazon SDK 或 CLI 配合使用
- 将 ListServerCertificates 与 Amazon SDK 或 CLI 配合使用
- 将 ListSigningCertificates 与 CLI 配合使用
- 将 ListUserPolicies 与 Amazon SDK 或 CLI 配合使用
- 将 ListUserTags 与 CLI 配合使用
- 将 ListUsers 与 Amazon SDK 或 CLI 配合使用
- 将 ListVirtualMfaDevices 与 CLI 配合使用
- 将 PutGroupPolicy 与 CLI 配合使用
- 将 PutRolePermissionsBoundary 与 CLI 配合使用
- 将 PutRolePolicy 与 Amazon SDK 或 CLI 配合使用
- 将 PutUserPermissionsBoundary 与 CLI 配合使用
- 将 PutUserPolicy 与 Amazon SDK 或 CLI 配合使用
- 将 RemoveClientIdFromOpenIdConnectProvider 与 CLI 配合使用
- 将 RemoveRoleFromInstanceProfile 与 CLI 配合使用
- 将 RemoveUserFromGroup 与 CLI 配合使用
- 将 ResyncMfaDevice 与 CLI 配合使用
- 将 SetDefaultPolicyVersion 与 Amazon SDK 或 CLI 配合使用
- 将 TagRole 与 CLI 配合使用
- 将 TagUser 与 CLI 配合使用
- 将 UntagRole 与 CLI 配合使用
- 将 UntagUser 与 CLI 配合使用
- 将 UpdateAccessKey 与 Amazon SDK 或 CLI 配合使用

- 将 UpdateAccountPasswordPolicy 与 CLI 配合使用
- 将 UpdateAssumeRolePolicy 与 CLI 配合使用
- 将 UpdateGroup 与 CLI 配合使用
- 将 UpdateLoginProfile 与 CLI 配合使用
- 将 UpdateOpenIdConnectProviderThumbprint 与 CLI 配合使用
- 将 UpdateRole 与 CLI 配合使用
- 将 UpdateRoleDescription 与 CLI 配合使用
- 将 UpdateSamlProvider 与 CLI 配合使用
- 将 UpdateServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UpdateSigningCertificate 与 CLI 配合使用
- 将 UpdateUser 与 Amazon SDK 或 CLI 配合使用
- 将 UploadServerCertificate 与 Amazon SDK 或 CLI 配合使用
- 将 UploadSigningCertificate 与 CLI 配合使用

将 **AddClientIdToOpenIdConnectProvider** 与 CLI 配合使用

以下代码示例演示如何使用 AddClientIdToOpenIdConnectProvider。

CLI

Amazon CLI

将客户端 ID（受众）添加到 Open-ID Connect（OIDC）提供者

以下 add-client-id-to-open-id-connect-provider 命令将客户端 ID my-application-ID 添加到名为 server.example.com 的 OIDC 提供者。

```
aws iam add-client-id-to-open-id-connect-provider \
    --client-id my-application-ID \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

此命令不生成任何输出。

要创建 OIDC 提供者，请使用 create-open-id-connect-provider 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AddClientIdToOpenIdConnectProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将客户端 ID（或受众）**my-application-ID** 添加到名为 **server.example.com** 的现有 OIDC 提供者。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AddClientIdToOpenIdConnectProvider。

Tools for PowerShell V5

示例 1：此命令将客户端 ID（或受众）**my-application-ID** 添加到名为 **server.example.com** 的现有 OIDC 提供者。

```
Add-IAMClientIDToOpenIDConnectProvider -ClientID "my-application-ID"
 -OpenIDConnectProviderARN "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 AddClientIdToOpenIdConnectProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AddRoleToInstanceProfile** 与 CLI 配合使用

以下代码示例演示如何使用 AddRoleToInstanceProfile。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 创建 EC2 启动类型的 Amazon ECS 服务
- Amazon MSK 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

CLI

Amazon CLI

为实例配置文件添加角色

以下 add-role-to-instance-profile 命令可将名为 S3Access 的角色添加到名为 Webserver 的实例配置文件。

```
aws iam add-role-to-instance-profile \
    --role-name S3Access \
    --instance-profile-name Webserver
```

此命令不生成任何输出。

要创建实例配置文件，请使用 create-instance-profile 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AddRoleToInstanceProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将名为 **S3Access** 的角色添加到名为 **webserver** 的现有实例配置文件中。要创建实例配置文件，请使用 **New-IAMInstanceProfile** 命令。使用此命令创建实例配置文件并将其与角色关联后，您可以将其附加到 EC2 实例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile-Name** 参数的 **New-EC2Instance** cmdlet 来启动新实例。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AddRoleToInstanceProfile。

Tools for PowerShell V5

示例 1：此命令将名为 **S3Access** 的角色添加到名为 **webserver** 的现有实例配置文件中。要创建实例配置文件，请使用 **New-IAMInstanceProfile** 命令。使用此命令创建实例配置文件并将其与角色关联后，您可以将其附加到 EC2 实例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile-Name** 参数的 **New-EC2Instance** cmdlet 来启动新实例。

```
Add-IAMRoleToInstanceProfile -RoleName "S3Access" -InstanceProfileName
 "webserver"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 AddRoleToInstanceProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AddUserToGroup** 与 CLI 配合使用

以下代码示例演示如何使用 AddUserToGroup。

CLI

Amazon CLI

将用户添加到 IAM 组

以下 add-user-to-group 命令可将名为 Bob 的 IAM 用户添加到名为 Admins 的 IAM 组。

```
aws iam add-user-to-group \
    --user-name Bob \
    --group-name Admins
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 用户组中添加和删除用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AddUserToGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令将名为 **Bob** 的用户添加到名为 **Admins** 的组中。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AddUserToGroup。

Tools for PowerShell V5

示例 1：此命令将名为 **Bob** 的用户添加到名为 **Admins** 的组中。

```
Add-IAMUserToGroup -UserName "Bob" -GroupName "Admins"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 AddUserToGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AttachGroupPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 `AttachGroupPolicy`。

CLI

Amazon CLI

将托管策略附加到 IAM 组

以下 `attach-group-policy` 命令可将名为 `ReadOnlyAccess` 的 Amazon 托管策略附加到名为 `Finance` 的 IAM 组。

```
aws iam attach-group-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --group-name Finance
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的托管策略和内联策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AttachGroupPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **TesterPolicy** 的客户管理型策略附加到 IAM 组 **Testers**。该组中的用户会立即受到该策略默认版本中所定义权限的影响。

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

示例 2：此示例将名为 **AdministratorAccess** 的 Amazon 托管策略附加到 IAM 组 **Admins**。该组中的用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AttachGroupPolicy。

Tools for PowerShell V5

示例 1：此示例将名为 **TesterPolicy** 的客户管理型策略附加到 IAM 组 **Testers**。该组中的用户会立即受到该策略默认版本中所定义权限的影响。

```
Register-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

示例 2：此示例将名为 **AdministratorAccess** 的 Amazon 托管策略附加到 IAM 组 **Admins**。该组中的用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMGroupPolicy -GroupName Admins -PolicyArn arn:aws:iam::aws:policy/
AdministratorAccess
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 AttachGroupPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`AttachRolePolicy`** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AttachRolePolicy`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 配置 Amazon ECS Service Connect
- 使用 Lambda 代理集成创建 REST API
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 EC2 启动类型的 Amazon ECS 服务
- 创建 Amazon Managed Grafana 工作区
- 创建第一个 Lambda 函数
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- IoT Device Defender 入门
- Step Functions 入门
- 管理角色
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数
- 处理 Streams 和生存时间

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Attach an IAM policy to a role.
    /// </summary>
    /// <param name="policyArn">The policy to attach.</param>
    /// <param name="roleName">The role that the policy will be attached to.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> AttachRolePolicyAsync(string policyArn, string
 roleName)
    {
        var response = await _IAMService.AttachRolePolicyAsync(new
 AttachRolePolicyRequest
        {
            PolicyArn = policyArn,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 AttachRolePolicy。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_attach_role_policy
#
# This function attaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_attach_role_policy() {
  local role_name policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_attach_role_policy"
    echo "Attaches an AWS Identity and Access Management (IAM) policy to an IAM
 role."
    echo "  -n role_name   The name of the IAM role."
```

```
  echo "  -p policy_ARN -- The IAM policy document ARN."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy ARN with the -p parameter."
  usage
  return 1
fi

response=$(aws iam attach-role-policy \
  --role-name "$role_name" \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports attach-role-policy operation failed.\n$response"
  return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 AttachRolePolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::attachRolePolicy(const Aws::String &roleName,
                                   const Aws::String &policyArn,
                                   const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::ListAttachedRolePoliciesRequest list_request;
    list_request.SetRoleName(roleName);

    bool done = false;
    while (!done) {
        auto list_outcome = iam.ListAttachedRolePolicies(list_request);
        if (!list_outcome.IsSuccess()) {
            std::cerr << "Failed to list attached policies of role " <<
                      roleName << ": " << list_outcome.GetError().GetMessage() <<
                      std::endl;
            return false;
        }

        const auto &policies = list_outcome.GetResult().GetAttachedPolicies();
        if (std::any_of(policies.cbegin(), policies.cend(),
                        [=](const Aws::IAM::Model::AttachedPolicy &policy) {
```

```
                                return policy.GetPolicyArn() == policyArn;
                            })) {
            std::cout << "Policy " << policyArn <<
                      " is already attached to role " << roleName << std::endl;
            return true;
        }

        done = !list_outcome.GetResult().GetIsTruncated();
        list_request.SetMarker(list_outcome.GetResult().GetMarker());
    }

    Aws::IAM::Model::AttachRolePolicyRequest request;
    request.SetRoleName(roleName);
    request.SetPolicyArn(policyArn);

    Aws::IAM::Model::AttachRolePolicyOutcome outcome =
 iam.AttachRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Failed to attach policy " << policyArn << " to role " <<
                  roleName << ": " << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        std::cout << "Successfully attached policy " << policyArn << " to role "
 <<
                  roleName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 AttachRolePolicy。

CLI

Amazon CLI

将托管策略附加到 IAM 角色

以下 `attach-role-policy` 命令可将名为 `ReadOnlyAccess` 的 Amazon 托管策略附加到名为 `ReadOnlyRole` 的 IAM 角色。

```
aws iam attach-role-policy \
    --policy-arn arn:aws:iam::aws:policy/ReadOnlyAccess \
    --role-name ReadOnlyRole
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的托管策略和内联策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AttachRolePolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// AttachRolePolicy attaches a policy to a role.
```

```
func (wrapper RoleWrapper) AttachRolePolicy(ctx context.Context, policyArn
 string, roleName string) error {
 _, err := wrapper.IamClient.AttachRolePolicy(ctx, &iam.AttachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't attach policy %v to role %v. Here's why: %v\n", policyArn,
 roleName, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 AttachRolePolicy。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.AttachRolePolicyRequest;
import software.amazon.awssdk.services.iam.model.AttachedPolicy;
import software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesRequest;
import
 software.amazon.awssdk.services.iam.model.ListAttachedRolePoliciesResponse;
import java.util.List;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
```

```
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AttachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleName = args[0];
        String policyArn = args[1];

        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        attachIAMRolePolicy(iam, roleName, policyArn);
        iam.close();
    }

    public static void attachIAMRolePolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            ListAttachedRolePoliciesRequest request =
 ListAttachedRolePoliciesRequest.builder()
                    .roleName(roleName)
                    .build();
```

```
            ListAttachedRolePoliciesResponse response =
 iam.listAttachedRolePolicies(request);
            List<AttachedPolicy> attachedPolicies = response.attachedPolicies();

            // Ensure that the policy is not attached to this role
            String polArn = "";
            for (AttachedPolicy policy : attachedPolicies) {
                polArn = policy.policyArn();
                if (polArn.compareTo(policyArn) == 0) {
                    System.out.println(roleName + " policy is already attached to
 this role.");
                    return;
                }
            }

            AttachRolePolicyRequest attachRequest =
 AttachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.attachRolePolicy(attachRequest);

            System.out.println("Successfully attached policy " + policyArn +
                    " to role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 AttachRolePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

附加策略。

```
import { AttachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const attachRolePolicy = (policyArn, roleName) => {
  const command = new AttachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-policies.html#iam-examples-policies-attaching-role-policy。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AttachRolePolicy。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        console.log(
          "AmazonDynamoDBFullAccess is already attached to this role."
        );
        process.exit();
      }
    });
    var params = {
      PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
      RoleName: process.argv[2],
    };
    iam.attachRolePolicy(params, function (err, data) {
      if (err) {
        console.log("Unable to attach policy to role", err);
      } else {
        console.log("Role attached successfully");
```

```
      }
    });
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-policies.html#iam-examples-policies-attaching-role-policy。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AttachRolePolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun attachIAMRolePolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        ListAttachedRolePoliciesRequest {
            roleName = roleNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAttachedRolePolicies(request)
        val attachedPolicies = response.attachedPolicies

        // Ensure that the policy is not attached to this role.
        val checkStatus: Int
        if (attachedPolicies != null) {
            checkStatus = checkList(attachedPolicies, policyArnVal)
            if (checkStatus == -1) {
                return
```

```
            }
        }

        val policyRequest =
            AttachRolePolicyRequest {
                roleName = roleNameVal
                policyArn = policyArnVal
            }
        iamClient.attachRolePolicy(policyRequest)
        println("Successfully attached policy $policyArnVal to role
 $roleNameVal")
    }
}

fun checkList(
    attachedPolicies: List<AttachedPolicy>,
    policyArnVal: String,
): Int {
    for (policy in attachedPolicies) {
        val polArn = policy.policyArn.toString()

        if (polArn.compareTo(policyArnVal) == 0) {
            println("The policy is already attached to this role.")
            return -1
        }
    }
    return 0
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 AttachRolePolicy。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

$service->attachRolePolicy($assumeRoleRole['RoleName'],
 $listAllBucketsPolicy['Arn']);

    public function attachRolePolicy($roleName, $policyArn)
    {
        return $this->customWaiter(function () use ($roleName, $policyArn) {
            $this->iamClient->attachRolePolicy([
                'PolicyArn' => $policyArn,
                'RoleName' => $roleName,
            ]);
        });
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 AttachRolePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **SecurityAudit** 的 Amazon 托管策略附加到 IAM 角色 **CoSecurityAuditors**。担任该角色的用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AttachRolePolicy。

Tools for PowerShell V5

示例 1：此示例将名为 **SecurityAudit** 的 Amazon 托管策略附加到 IAM 角色 **CoSecurityAuditors**。担任该角色的用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMRolePolicy -RoleName CoSecurityAuditors -PolicyArn
 arn:aws:iam::aws:policy/SecurityAudit
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 AttachRolePolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用 Boto3 策略对象将策略附加到角色。

```
def attach_to_role(role_name, policy_arn):
    """
    Attaches a policy to a role.
```

```
    :param role_name: The name of the role. **Note** this is the name, not the
ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).attach_role(RoleName=role_name)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
role_name)
        raise
```

使用 Boto3 角色对象将策略附加到角色。

```
def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 AttachRolePolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 AttachRolePolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn attach_role_policy(
    client: &iamClient,
    role: &Role,
    policy: &Policy,
) -> Result<AttachRolePolicyOutput, SdkError<AttachRolePolicyError>> {
    client
        .attach_role_policy()
        .role_name(role.role_name())
        .policy_arn(policy.arn().unwrap_or_default())
        .send()
        .await
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 AttachRolePolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    lo_iam->attachrolepolicy(
      iv_rolename = iv_role_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy attached to role successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role or policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 AttachRolePolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func attachRolePolicy(role: String, policyArn: String) async throws {
        let input = AttachRolePolicyInput(
            policyArn: policyArn,
            roleName: role
        )
        do {
            _ = try await client.attachRolePolicy(input: input)
        } catch {
            print("ERROR: Attaching a role policy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for St API 参考中的 AttachRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AttachUserPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 AttachUserPolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 创建只读和读写用户

CLI

Amazon CLI

将托管策略附加到 IAM 用户

以下 attach-user-policy 命令可将名为 AdministratorAccess 的 Amazon 托管策略附加到名为 Alice 的 IAM 用户。

```
aws iam attach-user-policy \
```

```
    --policy-arn arn:aws:iam::aws:policy/AdministratorAccess \
    --user-name Alice
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的托管策略和内联策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AttachUserPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **AmazonCognitoPowerUser** 的 Amazon 托管策略附加到 IAM 用户 **Bob**。用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AttachUserPolicy。

Tools for PowerShell V5

示例 1：此示例将名为 **AmazonCognitoPowerUser** 的 Amazon 托管策略附加到 IAM 用户 **Bob**。用户会立即受到该策略最新版本中所定义权限的影响。

```
Register-IAMUserPolicy -UserName Bob -PolicyArn arn:aws:iam::aws:policy/
AmazonCognitoPowerUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 AttachUserPolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 AttachUserPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Attaches a policy to a user
#
# @param user_name [String] The name of the user
# @param policy_arn [String] The Amazon Resource Name (ARN) of the policy
# @return [Boolean] true if successful, false otherwise
def attach_policy_to_user(user_name, policy_arn)
  @iam_client.attach_user_policy(
    user_name: user_name,
    policy_arn: policy_arn
  )
```

```
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to user: #{e.message}")
    false
  end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 AttachUserPolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn attach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .attach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 AttachUserPolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    lo_iam->attachuserpolicy(
      iv_username = iv_user_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy attached to user successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User or policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy attachment limit exceeded.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 AttachUserPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ChangePassword** 与 CLI 配合使用

以下代码示例演示如何使用 ChangePassword。

CLI

Amazon CLI

更改 IAM 用户的密码

要更改 IAM 用户的密码，建议使用 `--cli-input-json` 参数传递包含新旧密码的 JSON 文件。通过此方法，您可以使用含有非字母数字字符的强密码。将密码作为命令行参数传递时，

可能难以使用含有非字母数字字符的密码。要使用 `--cli-input-json` 参数，请先使用带 `--generate-cli-skeleton` 参数的 `change-password` 命令，如以下示例所示。

```
aws iam change-password \
    --generate-cli-skeleton > change-password.json
```

前述命令创建一个名为 change-password.json 的 JSON 文件，您可以用该文件来填写旧密码和新密码。例如，该文件可能如下所示。

```
{
    "OldPassword": "3s0K_;xh4~8XXI",
    "NewPassword": "]35d/{pB9Fo9wJ"
}
```

接下来，要更改密码，请再次使用 `change-password` 命令，这次是传递 `--cli-input-json` 参数以指定您的 JSON 文件。以下 `change-password` 命令将 `--cli-input-json` 参数与名为 change-password.json 的 JSON 文件一起使用。

```
aws iam change-password \
    --cli-input-json file://change-password.json
```

此命令不生成任何输出。

此命令只能由 IAM 用户调用。如果使用 Amazon 账户（根）凭证调用此命令，则命令将返回 `InvalidUserType` 错误。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 用户如何更改自己的密码。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ChangePassword。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令更改运行该命令的用户的密码。此命令只能由 IAM 用户调用。如果您在使用 Amazon 账户（根）凭证登录时调用此命令，则命令将返回 **`InvalidUserType`** 错误。

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ChangePassword。

Tools for PowerShell V5

示例 1：此命令更改运行该命令的用户的密码。此命令只能由 IAM 用户调用。如果您在使用 Amazon 账户（根）凭证登录时调用此命令，则命令将返回 **InvalidUserType** 错误。

```
Edit-IAMPassword -OldPassword "MyOldP@ssw0rd" -NewPassword "MyNewP@ssw0rd"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ChangePassword。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateAccessKey** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateAccessKey。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户
- 管理访问密钥

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Create an IAM access key for a user.
    /// </summary>
    /// <param name="userName">The username for which to create the IAM access
    /// key.</param>
```

```
    /// <returns>The AccessKey.</returns>
    public async Task<AccessKey> CreateAccessKeyAsync(string userName)
    {
        var response = await _IAMService.CreateAccessKeyAsync(new
 CreateAccessKeyRequest
        {
            UserName = userName,
        });

        return response.AccessKey;

    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 CreateAccessKey。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_create_user_access_key
#
# This function creates an IAM access key for the specified user.
```

```
#
# Parameters:
#       -u user_name -- The name of the IAM user.
#       [-f file_name] -- The optional file name for the access key output.
#
# Returns:
#       [access_key_id access_key_secret]
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user_access_key() {
  local user_name file_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) key pair."
    echo "  -u user_name   The name of the IAM user."
    echo "  [-f file_name]   Optional file name for the access key output."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:f:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      f) file_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
```

```
    usage
    return 1
  fi

  response=$(aws iam create-access-key \
    --user-name "$user_name" \
    --output text)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-access-key operation failed.$response"
    return 1
  fi

  if [[ -n "$file_name" ]]; then
    echo "$response" >"$file_name"
  fi

  local key_id key_secret
  # shellcheck disable=SC2086
  key_id=$(echo $response | cut -f 2 -d ' ')
  # shellcheck disable=SC2086
  key_secret=$(echo $response | cut -f 4 -d ' ')

  echo "$key_id $key_secret"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 CreateAccessKey。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
Aws::String AwsDoc::IAM::createAccessKey(const Aws::String &userName,
                                         const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateAccessKeyRequest request;
    request.SetUserName(userName);

    Aws::String result;
    Aws::IAM::Model::CreateAccessKeyOutcome outcome =
 iam.CreateAccessKey(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating access key for IAM user " << userName
                  << ":" << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &accessKey = outcome.GetResult().GetAccessKey();
        std::cout << "Successfully created access key for IAM user " <<
                  userName << std::endl << "  aws_access_key_id = " <<
                  accessKey.GetAccessKeyId() << std::endl <<
                  " aws_secret_access_key = " << accessKey.GetSecretAccessKey()
 <<
                  std::endl;
        result = accessKey.GetAccessKeyId();
    }

    return result;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 CreateAccessKey。

CLI

Amazon CLI

为 IAM 用户创建访问密钥

以下 `create-access-key` 命令创建名为 Bob IAM 用户的访问密钥（访问密钥 ID 和秘密访问密钥）。

```
aws iam create-access-key \
    --user-name Bob
```

输出：

```
{
    "AccessKey": {
        "UserName": "Bob",
        "Status": "Active",
        "CreateDate": "2015-03-09T18:39:23.411Z",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
    }
}
```

将秘密访问密钥存储在安全位置。如果访问密钥丢失，将无法恢复，必须创建一个新的访问密钥。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的访问密钥。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateAccessKey。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateAccessKeyPair creates an access key for a user. The returned access key
 contains
// the ID and secret credentials needed to use the key.
func (wrapper UserWrapper) CreateAccessKeyPair(ctx context.Context, userName
 string) (*types.AccessKey, error) {
 var key *types.AccessKey
 result, err := wrapper.IamClient.CreateAccessKey(ctx, &iam.CreateAccessKeyInput{
  UserName: aws.String(userName)})
 if err != nil {
  log.Printf("Couldn't create access key pair for user %v. Here's why: %v\n",
 userName, err)
 } else {
  key = result.AccessKey
 }
 return key, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 CreateAccessKey。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.CreateAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.CreateAccessKeyResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                   <user>\s

                Where:
                   user - An AWS IAM user that you can obtain from the AWS
 Management Console.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String user = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String keyId = createIAMAccessKey(iam, user);
        System.out.println("The Key Id is " + keyId);
        iam.close();
    }

    public static String createIAMAccessKey(IamClient iam, String user) {
        try {
            CreateAccessKeyRequest request = CreateAccessKeyRequest.builder()
                    .userName(user)
                    .build();

            CreateAccessKeyResponse response = iam.createAccessKey(request);
            return response.accessKey().accessKeyId();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 CreateAccessKey。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建访问密钥。

```
import { CreateAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 */
export const createAccessKey = (userName) => {
  const command = new CreateAccessKeyCommand({ UserName: userName });
  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateAccessKey。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccessKey({ UserName: "IAM_USER_NAME" }, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data.AccessKey);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateAccessKey。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun createIAMAccessKey(user: String?): String {
    val request =
        CreateAccessKeyRequest {
            userName = user
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createAccessKey(request)
        return response.accessKey?.accessKeyId.toString()
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 CreateAccessKey。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建新的访问密钥和秘密访问密钥对，并将其分配给用户 **David**。确保将 **AccessKeyId** 和 **SecretAccessKey** 值保存到文件中，因为这是您唯一可以获得 **SecretAccessKey** 的时间。您以后无法检索它。如果您丢失了秘密密钥，则必须创建一个新的访问密钥对。

```
New-IAMAccessKey -UserName David
```

输出：

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateAccessKey。

Tools for PowerShell V5

示例 1：此示例创建新的访问密钥和秘密访问密钥对，并将其分配给用户 **David**。确保将 **AccessKeyId** 和 **SecretAccessKey** 值保存到文件中，因为这是您唯一可以获得 **SecretAccessKey** 的时间。您以后无法检索它。如果您丢失了秘密密钥，则必须创建一个新的访问密钥对。

```
New-IAMAccessKey -UserName David
```

输出：

```
AccessKeyId     : AKIAIOSFODNN7EXAMPLE
CreateDate      : 4/13/2015 1:00:42 PM
SecretAccessKey : wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
Status          : Active
UserName        : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateAccessKey。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreateAccessKey。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
```

```
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
```

```
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 CreateAccessKey。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_access_key(client: &iamClient, user_name: &str) ->
 Result<AccessKey, iamError> {
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<CreateAccessKeyOutput, SdkError<CreateAccessKeyError>> =
 loop {
        match client.create_access_key().user_name(user_name).send().await {
            Ok(inner_response) => {
                break Ok(inner_response);
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
    };

    Ok(response.unwrap().access_key.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 CreateAccessKey。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->createaccesskey(
      iv_username = iv_user_name ).
    MESSAGE 'Access key created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Maximum number of access keys reached.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreateAccessKey。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func createAccessKey(userName: String) async throws ->
 IAMClientTypes.AccessKey {
        let input = CreateAccessKeyInput(
            userName: userName
        )
        do {
            let output = try await iamClient.createAccessKey(input: input)
            guard let accessKey = output.accessKey else {
                throw ServiceHandlerError.keyError
            }
            return accessKey
        } catch {
            print("ERROR: createAccessKey:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 CreateAccessKey。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateAccountAlias** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateAccountAlias。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::createAccountAlias(const Aws::String &aliasName,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::CreateAccountAliasRequest request;
    request.SetAccountAlias(aliasName);

    Aws::IAM::Model::CreateAccountAliasOutcome outcome = iam.CreateAccountAlias(
            request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating account alias " << aliasName << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created account alias " << aliasName <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 CreateAccountAlias。

CLI

Amazon CLI

创建账户别名

以下 create-account-alias 命令可为 Amazon 账户创建别名 examplecorp。

```
aws iam create-account-alias \
    --account-alias examplecorp
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的您的 Amazon 账户 ID 及其别名。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateAccountAlias。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.CreateAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateAccountAlias {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <alias>\s
```

```
                Where:
                    alias - The account alias to create (for example,
 myawsaccount).\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        createIAMAccountAlias(iam, alias);
        iam.close();
        System.out.println("Done");
    }

    public static void createIAMAccountAlias(IamClient iam, String alias) {
        try {
            CreateAccountAliasRequest request =
 CreateAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.createAccountAlias(request);
            System.out.println("Successfully created account alias: " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 CreateAccountAlias。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建账户别名。

```
import { CreateAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias - A unique name for the account alias.
 * @returns
 */
export const createAccountAlias = (alias) => {
  const command = new CreateAccountAliasCommand({
    AccountAlias: alias,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateAccountAlias。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.createAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateAccountAlias。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun createIAMAccountAlias(alias: String) {
    val request =
        CreateAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.createAccountAlias(request)
        println("Successfully created account alias named $alias")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 CreateAccountAlias。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将您 Amazon 账户的账户别名更改为 **mycompanyaws**。用户登录页面的地址将更改为 https://mycompanyaws.signin.aws.amazon.com/console。使用您的账户 ID 号而不是别名的原始 URL（https://<accountidnumber>.signin.aws.amazon.com/console）继续有效。但是，任何之前定义的、基于别名的 URL 都将停止工作。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 详细信息，请参阅《《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateAccountAlias。

Tools for PowerShell V5

示例 1：此示例将您 Amazon 账户的账户别名更改为 **mycompanyaws**。用户登录页面的地址将更改为 https://mycompanyaws.signin.aws.amazon.com/console。使用您的账户 ID 号而不是别名的原始 URL（https://<accountidnumber>.signin.aws.amazon.com/console）继续有效。但是，任何之前定义的、基于别名的 URL 都将停止工作。

```
New-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateAccountAlias。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreateAccountAlias。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 CreateAccountAlias。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->createaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Account alias already exists.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Account alias limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreateAccountAlias。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateGroup** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateGroup。

CLI

Amazon CLI

创建 IAM 组

以下 create-group 命令创建名为 Admins 的 IAM 组。

```
aws iam create-group \
    --group-name Admins
```

输出：

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-03-09T20:30:24.940Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateGroup。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { CreateGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const createGroup = async (groupName) => {
  const command = new CreateGroupCommand({ GroupName: groupName });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为 **Developers** 的新 IAM 组。

```
New-IAMGroup -GroupName Developers
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Developers
```

```
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateGroup。

Tools for PowerShell V5

示例 1：此示例创建了一个名为 **Developers** 的新 IAM 组。

```
New-IAMGroup -GroupName Developers
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 4/14/2015 11:21:31 AM
GroupId    : QNEJ5PM4NFSQCEXAMPLE1
GroupName  : Developers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 CreateGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateInstanceProfile** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateInstanceProfile`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 构建和管理弹性服务
- 创建 EC2 启动类型的 Amazon ECS 服务
- Amazon MSK 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   /// <summary>
   /// Create a policy, role, and profile that is associated with instances with
a specified name.
   /// An instance's associated profile defines a role that is assumed by the
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
                                        "]" +
                                        "}," +
```

```
                                "\"Action\": \"sts:AssumeRole\"" +
                                "}]" +
                            "}";

        var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

        var policyArn = "";

        try
        {
            var createPolicyResult = await _amazonIam.CreatePolicyAsync(
                new CreatePolicyRequest
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
```

```
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
```

```
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
        }
        return profileArn;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 CreateInstanceProfile。

CLI

Amazon CLI

创建实例配置文件

以下 `create-instance-profile` 命令创建名为 `Webserver` 的实例配置文件。

```
aws iam create-instance-profile \
    --instance-profile-name Webserver
```

输出：

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AIPAJMBYC7DLSPEXAMPLE",
        "Roles": [],
        "CreateDate": "2015-03-09T20:33:19.626Z",
        "InstanceProfileName": "Webserver",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:instance-profile/Webserver"
    }
```

```
}
```

要将角色添加到实例配置文件，请使用 `add-role-to-instance-profile` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateInstanceProfile。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const { InstanceProfile } = await iamClient.send(
  new CreateInstanceProfileCommand({
    InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
  }),
);
await waitUntilInstanceProfileExists(
  { client: iamClient },
  { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
);
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateInstanceProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个名为 **`ProfileForDevEC2Instance`** 的新 IAM 实例配置文件。您必须单独运行 **`Add-IAMRoleToInstanceProfile`** 命令，以将实例配置文件与为该实例提供权限的现有 IAM 角色相关联。最后，在启动 EC2 实例时将实例配置文件附加到该实

例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile_Name** 参数的 **New-EC2Instance** cmdlet。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateInstanceProfile。

Tools for PowerShell V5

示例 1：此示例创建了一个名为 **ProfileForDevEC2Instance** 的新 IAM 实例配置文件。您必须单独运行 **Add-IAMRoleToInstanceProfile** 命令，以将实例配置文件与为该实例提供权限的现有 IAM 角色相关联。最后，在启动 EC2 实例时将实例配置文件附加到该实例。为此，请使用带有 **InstanceProfile_Arn** 或 **InstanceProfile_Name** 参数的 **New-EC2Instance** cmdlet。

```
New-IAMInstanceProfile -InstanceProfileName ProfileForDevEC2Instance
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/
ProfileForDevEC2Instance
CreateDate          : 4/14/2015 11:31:39 AM
InstanceProfileId   : DYMFXL556EY46EXAMPLE1
InstanceProfileName : ProfileForDevEC2Instance
Path                : /
Roles               : {}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateInstanceProfile。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例将创建一个策略、角色和实例配置文件，并将其全部关联起来。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
```

```
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
created by
        this class. An instance's associated profile defines a role that is
assumed by the
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
```

```
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
```

```
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn
```

- 有关 API 详细信息，请参阅《适用于 Python（Boto3）的 Amazon SDK API 参考》中的 CreateInstanceProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateLoginProfile** 与 CLI 配合使用

以下代码示例演示如何使用 CreateLoginProfile。

CLI

Amazon CLI

为 IAM 用户创建密码

要为 IAM 用户创建密码，建议使用 --cli-input-json 参数传递包含密码的 JSON 文件。通过此方法，您可以创建含有非字母数字字符的强密码。将密码作为命令行参数传递时，可能难以创建含有非字母数字字符的密码。

要使用 --cli-input-json 参数，请先使用带 --generate-cli-skeleton 参数的 create-login-profile 命令，如以下示例所示。

```
aws iam create-login-profile \
    --generate-cli-skeleton > create-login-profile.json
```

前述命令创建了一个名为 create-login-profile.json 的 JSON 文件，您可以用该文件来填写后续 create-login-profile 命令的信息。例如：

```
{
    "UserName": "Bob",
    "Password": "&1-3a6u:RA0djs",
    "PasswordResetRequired": true
}
```

接下来，要为 IAM 用户创建密码，请再次使用 `create-login-profile` 命令，这次是传递 `--cli-input-json` 参数以指定您的 JSON 文件。以下 `create-login-profile` 命令将 `--cli-input-json` 参数与名为 create-login-profile.json 的 JSON 文件一起使用。

```
aws iam create-login-profile \
    --cli-input-json file://create-login-profile.json
```

输出：

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2015-03-10T20:55:40.274Z",
        "PasswordResetRequired": true
    }
}
```

如果新密码违反了账户密码策略，则该命令将返回 `PasswordPolicyViolation` 错误。

要更改已有密码用户的密码，请使用 `update-login-profile`。要为账户设置密码策略，请使用 `update-account-password-policy` 命令。

如果账户密码策略允许，则 IAM 用户可以使用 `change-password` 命令更改自己的密码。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的密码。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateLoginProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为名为 Bob 的 IAM 用户创建一个（临时）密码，并设置在 **Bob** 下次登录时要求该用户更改密码的标志。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

输出：

```
CreateDate                  PasswordResetRequired              UserName
----------                  ---------------------              --------
```

```
4/14/2015 12:26:30 PM         True                                  Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateLoginProfile。

Tools for PowerShell V5

示例 1：此示例为名为 Bob 的 IAM 用户创建一个（临时）密码，并设置在 **Bob** 下次登录时要求该用户更改密码的标志。

```
New-IAMLoginProfile -UserName Bob -Password P@ssw0rd -PasswordResetRequired $true
```

输出：

```
CreateDate                    PasswordResetRequired                 UserName
----------                    ---------------------                 --------
4/14/2015 12:26:30 PM         True                                  Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 CreateLoginProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateOpenIdConnectProvider** 与 CLI 配合使用

以下代码示例演示如何使用 CreateOpenIdConnectProvider。

CLI

Amazon CLI

创建 OpenID Connect（OIDC）提供者

要创建 OpenID Connect（OIDC）提供者，建议使用 --cli-input-json 参数来传递包含所需参数的 JSON 文件。创建 OIDC 提供者时，必须传递提供者的 URL，而且 URL 必须以 https:// 开头。将 URL 作为命令行参数传递可能很困难，因为冒号（:）和正斜杠（/）字符在某些命令行环境中具有特殊含义。使用 --cli-input-json 参数可以绕过这一限制。

要使用 --cli-input-json 参数，请先使用带 --generate-cli-skeleton 参数的 create-open-id-connect-provider 命令，如以下示例所示。

```
aws iam create-open-id-connect-provider \
    --generate-cli-skeleton > create-open-id-connect-provider.json
```

前述命令创建一个名为 create-open-id-connect-provider.json 的 JSON 文件，您可以用该文件来填写后续 `create-open-id-connect-provider` 命令的信息。例如：

```
{
    "Url": "https://server.example.com",
    "ClientIDList": [
        "example-application-ID"
    ],
    "ThumbprintList": [
        "c3768084dfb3d2b68b7897bf5f565da8eEXAMPLE"
    ]
}
```

如果您的 OIDC 提供者的发现和 JWKS 端点使用不同的证书，请在 `ThumbprintList` 中包含两个端点的指纹。

接下来，要创建 OpenID Connect（OIDC）提供者，请再次使用 `create-open-id-connect-provider` 命令，这次是传递 `--cli-input-json` 参数以指定您的 JSON 文件。以下 `create-open-id-connect-provider` 命令将 `--cli-input-json` 参数与名为 create-open-id-connect-provider.json 的 JSON 文件一起使用。

```
aws iam create-open-id-connect-provider \
    --cli-input-json file://create-open-id-connect-provider.json
```

输出：

```
{
    "OpenIDConnectProviderArn": "arn:aws:iam::123456789012:oidc-provider/
server.example.com"
}
```

有关 OIDC 提供者的更多信息，请参阅《Amazon IAM 用户指南》中的创建 OpenID Connect（OIDC）身份提供者。

有关获取 OIDC 提供者指纹的更多信息，请参阅《Amazon IAM 用户指南》中的获取 OpenID Connect 身份提供者的指纹。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateOpenIdConnectProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个 IAM OIDC 提供者，该提供者与位于 URL **https://example.oidcprovider.com** 中的 OIDC 兼容提供者服务和客户端 ID **my-testapp-1** 关联。OIDC 提供者将提供指纹。要验证指纹，请按照 http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html 中的步骤操作。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

输出：

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateOpenIdConnectProvider。

Tools for PowerShell V5

示例 1：此示例创建一个 IAM OIDC 提供者，该提供者与位于 URL **https://example.oidcprovider.com** 中的 OIDC 兼容提供者服务和客户端 ID **my-testapp-1** 关联。OIDC 提供者将提供指纹。要验证指纹，请按照 http://docs.aws.amazon.com/IAM/latest/UserGuide/identity-providers-oidc-obtain-thumbprint.html 中的步骤操作。

```
New-IAMOpenIDConnectProvider -Url https://example.oidcprovider.com -ClientIDList
 my-testapp-1 -ThumbprintList 990F419EXAMPLEECF12DDEDA5EXAMPLE52F20D9E
```

输出：

```
arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 CreateOpenIdConnectProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreatePolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreatePolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户
- 创建 Amazon Managed Grafana 工作区
- Amazon MSK 入门
- Step Functions 入门
- 管理策略
- 设置基于属性的访问控制
- 设置 Systems Manager
- 使用 IAM Policy Builder API

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Create an IAM policy.
    /// </summary>
    /// <param name="policyName">The name to give the new IAM policy.</param>
    /// <param name="policyDocument">The policy document for the new policy.</
param>
    /// <returns>The new IAM policy object.</returns>
    public async Task<ManagedPolicy> CreatePolicyAsync(string policyName, string
 policyDocument)
    {
```

```
        var response = await _IAMService.CreatePolicyAsync(new
 CreatePolicyRequest
        {
            PolicyDocument = policyDocument,
            PolicyName = policyName,
        });

        return response.Policy;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 CreatePolicy。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_policy
#
# This function creates an IAM policy.
#
# Parameters:
#       -n policy_name -- The name of the IAM policy.
#       -p policy_json -- The policy document.
#
```

```
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_policy() {
  local policy_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_policy"
    echo "Creates an AWS Identity and Access Management (IAM) policy."
    echo "  -n policy_name   The name of the IAM policy."
    echo "  -p policy_json -- The policy document."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:p:h" option; do
    case "${option}" in
      n) policy_name="${OPTARG}" ;;
      p) policy_document="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$policy_name" ]]; then
    errecho "ERROR: You must provide a policy name with the -n parameter."
    usage
    return 1
  fi

  if [[ -z "$policy_document" ]]; then
    errecho "ERROR: You must provide a policy document with the -p parameter."
    usage
```

```
    return 1
  fi

  response=$(aws iam create-policy \
    --policy-name "$policy_name" \
    --policy-document "$policy_document" \
    --output text \
    --query Policy.Arn)

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 CreatePolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
Aws::String AwsDoc::IAM::createPolicy(const Aws::String &policyName,
                                      const Aws::String &rsrcArn,
                                      const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreatePolicyRequest request;
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(BuildSamplePolicyDocument(rsrcArn));
```

```
    Aws::IAM::Model::CreatePolicyOutcome outcome = iam.CreatePolicy(request);
    Aws::String result;
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating policy " << policyName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        result = outcome.GetResult().GetPolicy().GetArn();
        std::cout << "Successfully created policy " << policyName <<
                  std::endl;
    }

    return result;
}

Aws::String AwsDoc::IAM::BuildSamplePolicyDocument(const Aws::String &rsrc_arn) {
    std::stringstream stringStream;
    stringStream << "{"
                 << "  \"Version\": \"2012-10-17\","
                 << "  \"Statement\": ["
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": \"logs:CreateLogGroup\","
                 << "        \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    },"
                 << "    {"
                 << "        \"Effect\": \"Allow\","
                 << "        \"Action\": ["
                 << "            \"dynamodb:DeleteItem\","
                 << "            \"dynamodb:GetItem\","
                 << "            \"dynamodb:PutItem\","
                 << "            \"dynamodb:Scan\","
                 << "            \"dynamodb:UpdateItem\""
                 << "       ],"
                 << "       \"Resource\": \""
                 << rsrc_arn
                 << "\""
                 << "    }"
                 << "   ]"
                 << "}";
```

```
    return stringStream.str();
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 CreatePolicy。

CLI

Amazon CLI

示例 1：创建客户托管策略

以下命令创建名为 my-policy 的客户托管策略。policy.json 文件是当前文件夹中的一个 JSON 文档，它授予对名为 amzn-s3-demo-bucket 的 Amazon S3 存储桶中 shared 文件夹的只读权限。

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json
```

policy.json 的内容：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:Get*",
                "s3:List*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/shared/*"
            ]
        }
    ]
}
```

输出：

```
{
```

```
    "Policy": {
        "PolicyName": "my-policy",
        "CreateDate": "2015-06-01T19:31:18.620Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "ZXR6A36LTYANPAI7NJ5UV",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::0123456789012:policy/my-policy",
        "UpdateDate": "2015-06-01T19:31:18.620Z"
    }
}
```

有关使用文件作为字符串参数输入的更多信息，请参阅《Amazon CLI 用户指南》中的为 Amazon CLI 指定参数值。

示例 2：创建带有描述的客户管理型策略

以下命令创建名为 my-policy 且带有不可变描述的客户管理型策略。

policy.json 文件是当前文件夹中的一个 JSON 文档，它授予对名为 amzn-s3-demo-bucket 的 Amazon S3 存储桶的所有 Put、List 和 Get 操作的访问权限。

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --description "This policy grants access to all Put, Get, and List actions
 for amzn-s3-demo-bucket"
```

policy.json 的内容：

```
{
   "Version":"2012-10-17",
   "Statement": [
       {
           "Effect": "Allow",
           "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
```

```
            ]
        }
    ]
}
```

输出：

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::123456789012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T22:38:47+00:00",
        "UpdateDate": "2023-05-24T22:38:47+00:00"
    }
}
```

有关基于身份的策略的更多信息，请参阅《Amazon IAM 用户指南》中的基于身份的策略和基于资源的策略。

示例 3：创建带有标签的客户管理型策略

以下命令创建名为 my-policy 且带有标签的客户托管策略。此示例使用带有以下 JSON 格式标签的 --tags 参数：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 参数可与简写格式的标签一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

policy.json 文件是当前文件夹中的一个 JSON 文档，它授予对名为 amzn-s3-demo-bucket 的 Amazon S3 存储桶的所有 Put、List 和 Get 操作的访问权限。

```
aws iam create-policy \
    --policy-name my-policy \
    --policy-document file://policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

policy.json 的内容：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

输出：

```
{
    "Policy": {
        "PolicyName": "my-policy",
        "PolicyId": "ANPAWGSUGIDPEXAMPLE",
        "Arn": "arn:aws:iam::12345678012:policy/my-policy",
        "Path": "/",
        "DefaultVersionId": "v1",
        "AttachmentCount": 0,
        "PermissionsBoundaryUsageCount": 0,
        "IsAttachable": true,
        "CreateDate": "2023-05-24T23:16:39+00:00",
        "UpdateDate": "2023-05-24T23:16:39+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
                "Key": "Location",
                "Value": "Seattle"
            {
        ]
    }
```

```
}
```

有关标记策略的更多信息，请参阅《Amazon IAM 用户指南》中的标记客户托管策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreatePolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// PolicyDocument defines a policy document as a Go struct that can be serialized
// to JSON.
type PolicyDocument struct {
 Version   string
 Statement []PolicyStatement
```

```
}

// PolicyStatement defines a statement in a policy document.
type PolicyStatement struct {
 Effect    string
 Action    []string
 Principal map[string]string `json:",omitempty"`
 Resource  *string           `json:",omitempty"`
}

// CreatePolicy creates a policy that grants a list of actions to the specified
 resource.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper PolicyWrapper) CreatePolicy(ctx context.Context, policyName string,
 actions []string,
 resourceArn string) (*types.Policy, error) {
 var policy *types.Policy
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(resourceArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n",
 resourceArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreatePolicy(ctx, &iam.CreatePolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
 })
 if err != nil {
  log.Printf("Couldn't create policy %v. Here's why: %v\n", policyName, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 CreatePolicy。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreatePolicyRequest;
import software.amazon.awssdk.services.iam.model.CreatePolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyRequest;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreatePolicy {

    public static final String PolicyDocument = "{" +
            "  \"Version\": \"2012-10-17\"," +
            "  \"Statement\": [" +
            "    {" +
            "        \"Effect\": \"Allow\"," +
            "        \"Action\": [" +
```

```
            "            \"dynamodb:DeleteItem\"," +
            "            \"dynamodb:GetItem\"," +
            "            \"dynamodb:PutItem\"," +
            "            \"dynamodb:Scan\"," +
            "            \"dynamodb:UpdateItem\"" +
            "       ]," +
            "       \"Resource\": \"*\"" +
            "    }" +
            "   ]" +
            "}";

    public static void main(String[] args) {

        final String usage = """
                Usage:
                    CreatePolicy <policyName>\s

                Where:
                    policyName - A unique policy name.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMPolicy(iam, policyName);
        System.out.println("Successfully created a policy with this ARN value: "
+ result);
        iam.close();
    }

    public static String createIAMPolicy(IamClient iam, String policyName) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreatePolicyRequest request = CreatePolicyRequest.builder()
```

```
                    .policyName(policyName)
                    .policyDocument(PolicyDocument)
                    .build();

            CreatePolicyResponse response = iam.createPolicy(request);

            // Wait until the policy is created.
            GetPolicyRequest polRequest = GetPolicyRequest.builder()
                    .policyArn(response.policy().arn())
                    .build();

            WaiterResponse<GetPolicyResponse> waitUntilPolicyExists =
 iamWaiter.waitUntilPolicyExists(polRequest);

 waitUntilPolicyExists.matched().response().ifPresent(System.out::println);
            return response.policy().arn();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 CreatePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建策略。

```
import { CreatePolicyCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} policyName
 */
export const createPolicy = (policyName) => {
  const command = new CreatePolicyCommand({
    PolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Action: "*",
          Resource: "*",
        },
      ],
    }),
    PolicyName: policyName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-policies.html#iam-examples-policies-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreatePolicy。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
```

```
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var myManagedPolicy = {
  Version: "2012-10-17",
  Statement: [
    {
      Effect: "Allow",
      Action: "logs:CreateLogGroup",
      Resource: "RESOURCE_ARN",
    },
    {
      Effect: "Allow",
      Action: [
        "dynamodb:DeleteItem",
        "dynamodb:GetItem",
        "dynamodb:PutItem",
        "dynamodb:Scan",
        "dynamodb:UpdateItem",
      ],
      Resource: "RESOURCE_ARN",
    },
  ],
};

var params = {
  PolicyDocument: JSON.stringify(myManagedPolicy),
  PolicyName: "myDynamoDBPolicy",
};

iam.createPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-policies.html#iam-examples-policies-creating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreatePolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun createIAMPolicy(policyNameVal: String?): String {
    val policyDocumentVal = """
    {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Action": [
                    "dynamodb:DeleteItem",
                    "dynamodb:GetItem",
                    "dynamodb:PutItem",
                    "dynamodb:Scan",
                    "dynamodb:UpdateItem"
                ],
                "Resource": "*"
            }
        ]
    }
    """.trimIndent()

    val request =
        CreatePolicyRequest {
            policyName = policyNameVal
            policyDocument = policyDocumentVal
        }
```

```
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createPolicy(request)
        return response.policy?.arn.toString()
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 CreatePolicy。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

$listAllBucketsPolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Action\": \"s3:ListAllMyBuckets\",
                    \"Resource\": \"arn:aws:s3:::*\"}]
}";
$listAllBucketsPolicy = $service->createPolicy("iam_demo_policy_$uuid",
 $listAllBucketsPolicyDocument);
echo "Created policy: {$listAllBucketsPolicy['PolicyName']}\n";

    /**
     * @param string $policyName
     * @param string $policyDocument
     * @return array
     */
    public function createPolicy(string $policyName, string $policyDocument)
    {
```

```
        $result = $this->customWaiter(function () use ($policyName,
$policyDocument) {
            return $this->iamClient->createPolicy([
                'PolicyName' => $policyName,
                'PolicyDocument' => $policyDocument,
            ]);
        });
        return $result['Policy'];
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 CreatePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在当前 Amazon 账户中创建一个名为 **MySamplePolicy** 的新 IAM 策略。文件 **MySamplePolicy.json** 提供策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

输出：

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreatePolicy。

Tools for PowerShell V5

示例 1：此示例在当前 Amazon 账户中创建一个名为 **MySamplePolicy** 的新 IAM 策略。文件 **MySamplePolicy.json** 提供策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicy -PolicyName MySamplePolicy -PolicyDocument (Get-Content -Raw
 MySamplePolicy.json)
```

输出：

```
Arn              : arn:aws:iam::123456789012:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 4/14/2015 2:45:59 PM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : LD4KP6HVFE7WGEXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 4/14/2015 2:45:59 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreatePolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.
```

```
    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
        raise
    else:
        return policy
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreatePolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 CreatePolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_policy(
    client: &iamClient,
    policy_name: &str,
    policy_document: &str,
) -> Result<Policy, iamError> {
    let policy = client
        .create_policy()
        .policy_name(policy_name)
        .policy_document(policy_document)
        .send()
        .await?;
    Ok(policy.policy.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 CreatePolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->createpolicy(
      iv_policyname = iv_policy_name
      iv_policydocument = iv_policy_document
      iv_description = iv_description ).
    MESSAGE 'Policy created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'Policy already exists.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.
    MESSAGE 'Policy document is malformed.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy limit exceeded.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreatePolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func createPolicy(name: String, policyDocument: String) async throws -
> IAMClientTypes.Policy {
        let input = CreatePolicyInput(
            policyDocument: policyDocument,
            policyName: name
        )
        do {
            let output = try await iamClient.createPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: createPolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 CreatePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreatePolicyVersion** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreatePolicyVersion。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理策略

CLI

Amazon CLI

创建新版本的托管策略

此示例创建了一个新的 v2 版本的 IAM policy，其 ARN 为 arn:aws:iam::123456789012:policy/MyPolicy，并设为默认版本。

```
aws iam create-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --policy-document file://NewPolicyVersion.json \
    --set-as-default
```

输出：

```
{
    "PolicyVersion": {
        "CreateDate": "2015-06-16T18:56:03.721Z",
        "VersionId": "v2",
        "IsDefaultVersion": true
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM policy 版本控制。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreatePolicyVersion。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建了一个新“v2”版本的 IAM 策略，其 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy**，并设为默认版本。**NewPolicyVersion.json** 文件提供了策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

输出：

```
CreateDate                          Document                 IsDefaultVersion
          VersionId
----------                          --------                 ----------------
          ---------
4/15/2015 10:54:54 AM                                        True
          v2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreatePolicyVersion。

Tools for PowerShell V5

示例 1：此示例创建了一个新“v2”版本的 IAM 策略，其 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy**，并设为默认版本。**NewPolicyVersion.json** 文件提供了策略内容。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。

```
New-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy -
PolicyDocument (Get-content -Raw NewPolicyVersion.json) -SetAsDefault $true
```

输出：

```
CreateDate                          Document                 IsDefaultVersion
          VersionId
----------                          --------                 ----------------
          ---------
```

```
4/15/2015 10:54:54 AM                                                  True
          v2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreatePolicyVersion。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
```

```
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreatePolicyVersion。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->createpolicyversion(
      iv_policyarn = iv_policy_arn
      iv_policydocument = iv_policy_document
      iv_setasdefault = iv_set_as_default ).
    MESSAGE 'Policy version created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Policy does not exist.' TYPE 'E'.
  CATCH /aws1/cx_iammalformedplydocex.
    MESSAGE 'Policy document is malformed.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Policy version limit exceeded.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreatePolicyVersion。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateRole** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateRole`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 配置 Amazon ECS Service Connect
- 使用 Lambda 代理集成创建 REST API
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 EC2 启动类型的 Amazon ECS 服务
- 创建 Amazon Managed Grafana 工作区
- 创建第一个 Lambda 函数
- Redshift Serverless 入门
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon Redshift 预置集群入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- IoT Device Defender 入门
- Step Functions 入门
- 管理角色
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数
- 处理 Streams 和生存时间

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Create a new IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="rolePolicyDocument">The name of the IAM policy document
    /// for the new role.</param>
    /// <returns>The Amazon Resource Name (ARN) of the role.</returns>
    public async Task<string> CreateRoleAsync(string roleName, string
rolePolicyDocument)
    {
        var request = new CreateRoleRequest
        {
            RoleName = roleName,
            AssumeRolePolicyDocument = rolePolicyDocument,
        };

        var response = await _IAMService.CreateRoleAsync(request);
        return response.Role.Arn;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 CreateRole。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_create_role
#
# This function creates an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_json -- The assume role policy document.
#
# Returns:
#       The ARN of the role.
#     And:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_create_role() {
  local role_name policy_document response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user_access_key"
    echo "Creates an AWS Identity and Access Management (IAM) role."
```

```
  echo "  -n role_name   The name of the IAM role."
  echo "  -p policy_json -- The assume role policy document."
  echo ""
}

# Retrieve the calling parameters.
while getopts "n:p:h" option; do
  case "${option}" in
    n) role_name="${OPTARG}" ;;
    p) policy_document="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$role_name" ]]; then
  errecho "ERROR: You must provide a role name with the -n parameter."
  usage
  return 1
fi

if [[ -z "$policy_document" ]]; then
  errecho "ERROR: You must provide a policy document with the -p parameter."
  usage
  return 1
fi

response=$(aws iam create-role \
  --role-name "$role_name" \
  --assume-role-policy-document "$policy_document" \
  --output text \
  --query Role.Arn)

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
```

```
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 CreateRole。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::createIamRole(
        const Aws::String &roleName,
        const Aws::String &policy,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient client(clientConfig);
    Aws::IAM::Model::CreateRoleRequest request;

    request.SetRoleName(roleName);
    request.SetAssumeRolePolicyDocument(policy);

    Aws::IAM::Model::CreateRoleOutcome outcome = client.CreateRole(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error creating role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const Aws::IAM::Model::Role iamRole = outcome.GetResult().GetRole();
        std::cout << "Created role " << iamRole.GetRoleName() << "\n";
        std::cout << "ID: " << iamRole.GetRoleId() << "\n";
```

```
        std::cout << "ARN: " << iamRole.GetArn() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 CreateRole。

CLI

Amazon CLI

示例 1：创建 IAM 角色

以下 `create-role` 命令创建一个名为 `Test-Role` 的角色并对其附加信任策略。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json
```

输出：

```
{
    "Role": {
        "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "CreateDate": "2013-06-07T20:43:32.821Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

信任策略在 Test-Role-Trust-Policy.json 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）信任策略必须指定主体。

要将权限策略附加到角色，请使用 `put-role-policy` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

示例 2：创建具有指定最长会话持续时间的 IAM 角色

以下 create-role 命令创建一个名为 Test-Role 的角色，并将最长会话持续时间设置为 7200 秒（2 小时）。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --max-session-duration 7200
```

输出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:role/Test-Role",
        "CreateDate": "2023-05-24T23:50:25+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::12345678012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        }
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色最长会话持续时间（Amazon API）。

示例 3：创建带有标签的 IAM 角色

以下命令创建带有标签的 IAM 角色。Test-Role此示例使用带有以下 JSON 格式标签的 --tags 参数标志：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 标志可与简写格式的标签一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-role \
    --role-name Test-Role \
    --assume-role-policy-document file://Test-Role-Trust-Policy.json \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

输出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "Test-Role",
        "RoleId": "AKIAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/Test-Role",
        "CreateDate": "2023-05-25T23:29:41+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Sid": "Statement1",
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole"
                }
            ]
        },
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateRole。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateRole creates a role that trusts a specified user. The trusted user can
 assume
// the role to acquire its permissions.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper RoleWrapper) CreateRole(ctx context.Context, roleName string,
 trustedUserArn string) (*types.Role, error) {
 var role *types.Role
 trustPolicy := PolicyDocument{
```

```
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:    "Allow",
   Principal: map[string]string{"AWS": trustedUserArn},
   Action:    []string{"sts:AssumeRole"},
  }},
 }
 policyBytes, err := json.Marshal(trustPolicy)
 if err != nil {
  log.Printf("Couldn't create trust policy for %v. Here's why: %v\n",
 trustedUserArn, err)
  return nil, err
 }
 result, err := wrapper.IamClient.CreateRole(ctx, &iam.CreateRoleInput{
  AssumeRolePolicyDocument: aws.String(string(policyBytes)),
  RoleName:                 aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't create role %v. Here's why: %v\n", roleName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 CreateRole。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import org.json.simple.JSONObject;
import org.json.simple.parser.JSONParser;
import software.amazon.awssdk.services.iam.model.CreateRoleRequest;
```

```
import software.amazon.awssdk.services.iam.model.CreateRoleResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import java.io.FileReader;

/*
*   This example requires a trust policy document. For more information, see:
*   https://aws.amazon.com/blogs/security/how-to-use-trust-policies-with-iam-
roles/
*
*
*  In addition, set up your development environment, including your credentials.
*
*  For information, see this documentation topic:
*
*  https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */

public class CreateRole {
    public static void main(String[] args) throws Exception {
        final String usage = """
                Usage:
                    <rolename> <fileLocation>\s

                Where:
                    rolename - The name of the role to create.\s
                    fileLocation - The location of the JSON document that
 represents the trust policy.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String rolename = args[0];
        String fileLocation = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
```

```
        String result = createIAMRole(iam, rolename, fileLocation);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMRole(IamClient iam, String rolename, String
 fileLocation) throws Exception {
        try {
            JSONObject jsonObject = (JSONObject)
 readJsonSimpleDemo(fileLocation);
            CreateRoleRequest request = CreateRoleRequest.builder()
                    .roleName(rolename)
                    .assumeRolePolicyDocument(jsonObject.toJSONString())
                    .description("Created using the AWS SDK for Java")
                    .build();

            CreateRoleResponse response = iam.createRole(request);
            System.out.println("The ARN of the role is " +
 response.role().arn());

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }

    public static Object readJsonSimpleDemo(String filename) throws Exception {
        FileReader reader = new FileReader(filename);
        JSONParser jsonParser = new JSONParser();
        return jsonParser.parse(reader);
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 CreateRole。

JavaScript

SDK for JavaScript（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建角色。

```
import { CreateRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const createRole = (roleName) => {
  const command = new CreateRoleCommand({
    AssumeRolePolicyDocument: JSON.stringify({
      Version: "2012-10-17",
      Statement: [
        {
          Effect: "Allow",
          Principal: {
            Service: "lambda.amazonaws.com",
          },
          Action: "sts:AssumeRole",
        },
      ],
    }),
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateRole。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

$assumeRolePolicyDocument = "{
                \"Version\": \"2012-10-17\",
                \"Statement\": [{
                    \"Effect\": \"Allow\",
                    \"Principal\": {\"AWS\": \"{$user['Arn']}\"},
                    \"Action\": \"sts:AssumeRole\"
                }]
            }";
$assumeRoleRole = $service->createRole("iam_demo_role_$uuid",
 $assumeRolePolicyDocument);
echo "Created role: {$assumeRoleRole['RoleName']}\n";

    /**
     * @param string $roleName
     * @param string $rolePolicyDocument
     * @return array
     * @throws AwsException
     */
    public function createRole(string $roleName, string $rolePolicyDocument)
    {
        $result = $this->customWaiter(function () use ($roleName,
 $rolePolicyDocument) {
            return $this->iamClient->createRole([
                'AssumeRolePolicyDocument' => $rolePolicyDocument,
                'RoleName' => $roleName,
```

```
            ]);
        });
        return $result['Role'];
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 CreateRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **MyNewRole** 的新角色，并将文件 **NewRoleTrustPolicy.json** 中的策略附加到该角色。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。输出中显示的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

输出：

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
                           %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                           %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateRole。

Tools for PowerShell V5

示例 1：此示例创建一个名为 **MyNewRole** 的新角色，并将文件 **NewRoleTrustPolicy.json** 中的策略附加到该角色。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 策略文件。输出中显示的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = New-IAMRole -AssumeRolePolicyDocument (Get-Content -raw
 NewRoleTrustPolicy.json) -RoleName MyNewRole
$results
```

输出：

```
Arn                       : arn:aws:iam::123456789012:role/MyNewRole
AssumeRolePolicyDocument : %7B%0D%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C
%0D%0A%20%20%22Statement%22
                           %3A%20%5B%0D%0A%20%20%20%20%7B%0D%0A
%20%20%20%20%20%20%22Sid%22%3A%20%22%22%2C
                           %0D%0A%20%20%20%20%20%20%22Effect%22%3A%20%22Allow
%22%2C%0D%0A%20%20%20%20%20%20
                           %22Principal%22%3A%20%7B%0D%0A
%20%20%20%20%20%20%20%20%22AWS%22%3A%20%22arn%3Aaws
                           %3Aiam%3A%3A123456789012%3ADavid%22%0D%0A
%20%20%20%20%20%20%7D%2C%0D%0A%20%20%20
```

```
                              %20%20%20%22Action%22%3A%20%22sts%3AAssumeRole%22%0D
%0A%20%20%20%20%7D%0D%0A%20
                              %20%5D%0D%0A%7D
CreateDate                : 4/15/2015 11:04:23 AM
Path                      : /
RoleId                    : V5PAJI2KPN4EAEXAMPLE1
RoleName                  : MyNewRole

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::123456789012:David"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.
```

```
    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role
```

- 有关 API 详细信息，请参阅 Amazon SDK for Python（Boto3）API Reference中的 CreateRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Creates a role and attaches policies to it.
 #
 # @param role_name [String] The name of the role.
 # @param assume_role_policy_document [Hash] The trust relationship policy
document.
 # @param policy_arns [Array<String>] The ARNs of the policies to attach.
 # @return [String, nil] The ARN of the new role if successful, or nil if an
error occurred.
 def create_role(role_name, assume_role_policy_document, policy_arns)
   response = @iam_client.create_role(
     role_name: role_name,
     assume_role_policy_document: assume_role_policy_document.to_json
   )
   role_arn = response.role.arn

   policy_arns.each do |policy_arn|
     @iam_client.attach_role_policy(
       role_name: role_name,
       policy_arn: policy_arn
     )
   end

   role_arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating role: #{e.message}")
   nil
 end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 CreateRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_role(
    client: &iamClient,
    role_name: &str,
    role_policy_document: &str,
) -> Result<Role, iamError> {
    let response: CreateRoleOutput = loop {
        if let Ok(response) = client
            .create_role()
            .role_name(role_name)
            .assume_role_policy_document(role_policy_document)
            .send()
            .await
        {
            break response;
        }
    };

    Ok(response.role.unwrap())
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Rust API Reference中的 CreateRole。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->createrole(
          iv_rolename = iv_role_name
          iv_assumerolepolicydocument = iv_assume_role_policy_document ).
        MESSAGE 'Role created successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'Role already exists.' TYPE 'E'.
      CATCH /aws1/cx_iammalformedplydocex.
```

```
        MESSAGE 'Assume role policy document is malformed.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Role limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreateRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func createRole(name: String, policyDocument: String) async throws ->
 String {
        let input = CreateRoleInput(
            assumeRolePolicyDocument: policyDocument,
            roleName: name
        )
        do {
            let output = try await client.createRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            guard let id = role.roleId else {
                throw ServiceHandlerError.noSuchRole
            }
            return id
        } catch {
            print("ERROR: createRole:", dump(error))
            throw error
```

```
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 CreateRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateSAMLProvider** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateSAMLProvider。

CLI

Amazon CLI

创建 SAML 提供者

此示例在 IAM 中创建了一个名为 MySAMLProvider 的新 SAML 提供者。SAMLMetaData.xml 文件中的 SAML 元数据文档对其进行了描述。

```
aws iam create-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --name MySAMLProvider
```

输出：

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/MySAMLProvider"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM SAML 身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateSAMLProvider。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { CreateSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import * as path from "node:path";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";

const client = new IAMClient({});

/**
 * This sample document was generated using Auth0.
 * For more information on generating this document,
 see https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_providers_create_saml.html#samlstep1.
 */
const sampleMetadataDocument = readFileSync(
  path.join(
    dirnameFromMetaUrl(import.meta.url),
    "../../../../resources/sample_files/sample_saml_metadata.xml",
  ),
);

/**
 *
 * @param {*} providerName
 * @returns
 */
export const createSAMLProvider = async (providerName) => {
  const command = new CreateSAMLProviderCommand({
    Name: providerName,
    SAMLMetadataDocument: sampleMetadataDocument.toString(),
  });

  const response = await client.send(command);
```

```
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateSAMLProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在 IAM 中创建了一个新 SAML 提供者实体。其命名为 **MySAMLProvider**，通过文件 **SAMLMetaData.xml** 中的 SAML 元数据文档描述，该文件从 SAML 服务提供者网站单独下载。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

输出：

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateSAMLProvider。

Tools for PowerShell V5

示例 1：此示例在 IAM 中创建了一个新 SAML 提供者实体。其命名为 **MySAMLProvider**，通过文件 **SAMLMetaData.xml** 中的 SAML 元数据文档描述，该文件从 SAML 服务提供者网站单独下载。

```
New-IAMSAMLProvider -Name MySAMLProvider -SAMLMetadataDocument (Get-Content -Raw
 SAMLMetaData.xml)
```

输出：

```
arn:aws:iam::123456789012:saml-provider/MySAMLProvider
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 CreateSAMLProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateServiceLinkedRole** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 CreateServiceLinkedRole。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Create an IAM service-linked role.
    /// </summary>
    /// <param name="serviceName">The name of the AWS Service.</param>
    /// <param name="description">A description of the IAM service-linked role.</
param>
    /// <returns>The IAM role that was created.</returns>
    public async Task<Role> CreateServiceLinkedRoleAsync(string serviceName,
 string description)
    {
        var request = new CreateServiceLinkedRoleRequest
        {
            AWSServiceName = serviceName,
            Description = description
        };

        var response = await _IAMService.CreateServiceLinkedRoleAsync(request);
        return response.Role;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 CreateServiceLinkedRole。

CLI

Amazon CLI

创建服务相关角色

以下 create-service-linked-role 示例为指定的 Amazon 服务创建了服务相关角色，并附加了指定的描述。

```
aws iam create-service-linked-role \
    --aws-service-name lex.amazonaws.com \
    --description "My service-linked role to support Lex"
```

输出：

```
{
    "Role": {
        "Path": "/aws-service-role/lex.amazonaws.com/",
        "RoleName": "AWSServiceRoleForLexBots",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::1234567890:role/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots",
        "CreateDate": "2019-04-17T20:34:14+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Action": [
                        "sts:AssumeRole"
                    ],
                    "Effect": "Allow",
                    "Principal": {
                        "Service": [
                            "lex.amazonaws.com"
                        ]
                    }
                }
            ]
        }
```

```
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用服务相关角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateServiceLinkedRole。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// CreateServiceLinkedRole creates a service-linked role that is owned by the
 specified service.
func (wrapper RoleWrapper) CreateServiceLinkedRole(ctx context.Context,
 serviceName string, description string) (
 *types.Role, error) {
```

```
 var role *types.Role
 result, err := wrapper.IamClient.CreateServiceLinkedRole(ctx,
 &iam.CreateServiceLinkedRoleInput{
  AWSServiceName: aws.String(serviceName),
  Description:    aws.String(description),
 })
 if err != nil {
  log.Printf("Couldn't create service-linked role %v. Here's why: %v\n",
 serviceName, err)
 } else {
  role = result.Role
 }
 return role, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 CreateServiceLinkedRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建服务相关角色。

```
import {
  CreateServiceLinkedRoleCommand,
  GetRoleCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
```

```
 *
 * @param {string} serviceName
 */
export const createServiceLinkedRole = async (serviceName) => {
  const command = new CreateServiceLinkedRoleCommand({
    // For a list of AWS services that support service-linked roles,
    // see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_aws-
services-that-work-with-iam.html.
    //
    // For a list of AWS service endpoints, see https://docs.aws.amazon.com/
general/latest/gr/aws-service-information.html.
    AWSServiceName: serviceName,
  });
  try {
    const response = await client.send(command);
    console.log(response);
    return response;
  } catch (caught) {
    if (
      caught instanceof Error &&
      caught.name === "InvalidInputException" &&
      caught.message.includes(
        "Service role name AWSServiceRoleForElasticBeanstalk has been taken in
 this account",
      )
    ) {
      console.warn(caught.message);
      return client.send(
        new GetRoleCommand({ RoleName: "AWSServiceRoleForElasticBeanstalk" }),
      );
    }
    throw caught;
  }
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateServiceLinkedRole。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function createServiceLinkedRole($awsServiceName, $customSuffix = "",
 $description = "")
    {
        $createServiceLinkedRoleArguments = ['AWSServiceName' =>
 $awsServiceName];
        if ($customSuffix) {
            $createServiceLinkedRoleArguments['CustomSuffix'] = $customSuffix;
        }
        if ($description) {
            $createServiceLinkedRoleArguments['Description'] = $description;
        }
        return $this->iamClient-
>createServiceLinkedRole($createServiceLinkedRoleArguments);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 CreateServiceLinkedRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为自动扩缩服务创建 servicelinked 角色。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateServiceLinkedRole。

Tools for PowerShell V5

示例 1：此示例为自动扩缩服务创建 servicelinked 角色。

```
New-IAMServiceLinkedRole -AWSServiceName autoscaling.amazonaws.com -CustomSuffix
 RoleNameEndsWithThis -Description "My service-linked role to support
 autoscaling"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateServiceLinkedRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_service_linked_role(service_name, description):
    """
    Creates a service-linked role.

    :param service_name: The name of the service that owns the role.
    :param description: A description to give the role.
    :return: The newly created role.
    """
    try:
        response = iam.meta.client.create_service_linked_role(
            AWSServiceName=service_name, Description=description
        )
        role = iam.Role(response["Role"]["RoleName"])
        logger.info("Created service-linked role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create service-linked role for %s.",
 service_name)
        raise
```

```
    else:
        return role
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreateServiceLinkedRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Creates a service-linked role
 #
 # @param service_name [String] The service name to create the role for.
 # @param description [String] The description of the service-linked role.
 # @param suffix [String] Suffix for customizing role name.
 # @return [String] The name of the created role
 def create_service_linked_role(service_name, description, suffix)
   response = @iam_client.create_service_linked_role(
     aws_service_name: service_name, description: description, custom_suffix:
suffix
   )
   role_name = response.role.role_name
   @logger.info("Created service-linked role #{role_name}.")
   role_name
 rescue Aws::Errors::ServiceError => e
   @logger.error("Couldn't create service-linked role for #{service_name}.
Here's why:")
   @logger.error("\t#{e.code}: #{e.message}")
   raise
 end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 CreateServiceLinkedRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_service_linked_role(
    client: &iamClient,
    aws_service_name: String,
    custom_suffix: Option<String>,
    description: Option<String>,
) -> Result<CreateServiceLinkedRoleOutput,
 SdkError<CreateServiceLinkedRoleError>> {
    let response = client
        .create_service_linked_role()
        .aws_service_name(aws_service_name)
        .set_custom_suffix(custom_suffix)
        .set_description(description)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 CreateServiceLinkedRole。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreateServiceLinkedRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func createServiceLinkedRole(service: String, suffix: String? = nil,
 description: String?)
```

```
                async throws -> IAMClientTypes.Role {
        let input = CreateServiceLinkedRoleInput(
            awsServiceName: service,
            customSuffix: suffix,
            description: description
        )
        do {
            let output = try await client.createServiceLinkedRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: createServiceLinkedRole:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 CreateServiceLinkedRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`CreateUser`** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateUser`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Create an IAM user.
    /// </summary>
    /// <param name="userName">The username for the new IAM user.</param>
    /// <returns>The IAM user that was created.</returns>
    public async Task<User> CreateUserAsync(string userName)
    {
        var response = await _IAMService.CreateUserAsync(new CreateUserRequest
{ UserName = userName });
        return response.User;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 CreateUser。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
###############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
```

```
# the global variable $VERBOSE is set to true.
###############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

###############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
###############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

###############################################################################
# function iam_create_user
#
# This function creates the specified IAM user, unless
# it already exists.
#
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       The ARN of the user.
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_create_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_create_user"
    echo "Creates an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user. It must be unique within the
 account."
    echo ""
```

```
}

# Retrieve the calling parameters.
while getopts "u:h" option; do
  case "${option}" in
    u) user_name="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$user_name" ]]; then
  errecho "ERROR: You must provide a username with the -u parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    User name:   $user_name"
iecho ""

# If the user already exists, we don't want to try to create it.
if (iam_user_exists "$user_name"); then
  errecho "ERROR: A user with that name already exists in the account."
  return 1
fi

response=$(aws iam create-user --user-name "$user_name" \
  --output text \
  --query 'User.Arn')

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports create-user operation failed.$response"
```

```
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 CreateUser。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::CreateUserRequest create_request;
    create_request.SetUserName(userName);

    auto create_outcome = iam.CreateUser(create_request);
    if (!create_outcome.IsSuccess()) {
        std::cerr << "Error creating IAM user " << userName << ":" <<
                  create_outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully created IAM user " << userName << std::endl;
    }

    return create_outcome.IsSuccess();
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 CreateUser。

CLI

Amazon CLI

示例 1：创建 IAM 用户

以下 `create-user` 命令可在当前账户中创建一个名为 Bob 的 IAM 用户。

```
aws iam create-user \
    --user-name Bob
```

输出：

```
{
    "User": {
        "UserName": "Bob",
        "Path": "/",
        "CreateDate": "2023-06-08T03:20:41.270Z",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Bob"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 账户中创建 IAM 用户。

示例 2：在指定路径创建 IAM 用户

以下 `create-user` 命令可在指定路径创建一个名为 Bob 的 IAM 用户。

```
aws iam create-user \
    --user-name Bob \
    --path /division_abc/subdivision_xyz/
```

输出：

```
{
    "User": {
        "Path": "/division_abc/subdivision_xyz/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/division_abc/subdivision_xyz/Bob",
        "CreateDate": "2023-05-24T18:20:17+00:00"
```

```
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 标识符。

示例 3：创建带有标签的 IAM 用户

以下 create-user 命令创建一个名为 Bob 且带有标签的 IAM 用户。此示例使用带有以下 JSON 格式标签的 --tags 参数标志：'{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location", "Value": "Seattle"}'。或者，--tags 标志可与简写格式的标签一起使用：'Key=Department,Value=Accounting Key=Location,Value=Seattle'。

```
aws iam create-user \
    --user-name Bob \
    --tags '{"Key": "Department", "Value": "Accounting"}' '{"Key": "Location",
 "Value": "Seattle"}'
```

输出：

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-25T17:14:21+00:00",
        "Tags": [
            {
                "Key": "Department",
                "Value": "Accounting"
            },
            {
                "Key": "Location",
                "Value": "Seattle"
            }
        ]
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 用户。

示例 3：创建具有设定权限边界的 IAM 用户

以下 `create-user` 命令创建一个名为 Bob 且权限边界为 AmazonS3FullAccess 的 IAM 用户。

```
aws iam create-user \
    --user-name Bob \
    --permissions-boundary arn:aws:iam::aws:policy/AmazonS3FullAccess
```

输出：

```
{
    "User": {
        "Path": "/",
        "UserName": "Bob",
        "UserId": "AIDAIOSFODNN7EXAMPLE",
        "Arn": "arn:aws:iam::12345678012:user/Bob",
        "CreateDate": "2023-05-24T17:50:53+00:00",
        "PermissionsBoundary": {
        "PermissionsBoundaryType": "Policy",
        "PermissionsBoundaryArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
        }
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 实体的权限边界。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateUser。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUser creates a new user with the specified name.
func (wrapper UserWrapper) CreateUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.CreateUser(ctx, &iam.CreateUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
 } else {
  user = result.User
 }
 return user, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 CreateUser。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.core.waiters.WaiterResponse;
import software.amazon.awssdk.services.iam.model.CreateUserRequest;
import software.amazon.awssdk.services.iam.model.CreateUserResponse;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.waiters.IamWaiter;
import software.amazon.awssdk.services.iam.model.GetUserRequest;
import software.amazon.awssdk.services.iam.model.GetUserResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username>\s

                Where:
                    username - The name of the user to create.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
```

```
            System.exit(1);
        }

        String username = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        String result = createIAMUser(iam, username);
        System.out.println("Successfully created user: " + result);
        iam.close();
    }

    public static String createIAMUser(IamClient iam, String username) {
        try {
            // Create an IamWaiter object.
            IamWaiter iamWaiter = iam.waiter();

            CreateUserRequest request = CreateUserRequest.builder()
                    .userName(username)
                    .build();

            CreateUserResponse response = iam.createUser(request);

            // Wait until the user is created.
            GetUserRequest userRequest = GetUserRequest.builder()
                    .userName(response.user().userName())
                    .build();

            WaiterResponse<GetUserResponse> waitUntilUserExists =
 iamWaiter.waitUntilUserExists(userRequest);

 waitUntilUserExists.matched().response().ifPresent(System.out::println);
            return response.user().userName();

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 CreateUser。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建 用户。

```
import { CreateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const createUser = (name) => {
  const command = new CreateUserCommand({ UserName: name });
  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-creating-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateUser。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    iam.createUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  } else {
    console.log(
      "User " + process.argv[2] + " already exists",
      data.User.UserId
    );
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-creating-users。

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 CreateUser。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun createIAMUser(usernameVal: String?): String? {
    val request =
        CreateUserRequest {
            userName = usernameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.createUser(request)
        return response.user?.userName
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 CreateUser。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
```

```
$service = new IAMService();

$user = $service->createUser("iam_demo_user_$uuid");
echo "Created user with the arn: {$user['Arn']}\n";


    /**
     * @param string $name
     * @return array
     * @throws AwsException
     */
    public function createUser(string $name): array
    {
        $result = $this->iamClient->createUser([
            'UserName' => $name,
        ]);

        return $result['User'];
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 CreateUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **Bob** 的 IAM 用户。如果 Bob 需要登录 Amazon 控制台，则必须单独运行命令 **New-IAMLoginProfile** 来创建带有密码的登录配置文件。如果 Bob 需要运行 Amazon PowerShell 或跨平台 CLI 命令或者 Amazon 进行 API 调用，则必须单独运行 **New-IAMAccessKey** 命令来创建访问密钥。

```
New-IAMUser -UserName Bob
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 4/22/2015 12:02:11 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : AIDAJWGEFDMEMEXAMPLE1
```

```
UserName          : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateUser。

Tools for PowerShell V5

示例 1：此示例创建一个名为 **Bob** 的 IAM 用户。如果 Bob 需要登录 Amazon 控制台，则必须单独运行命令 **New-IAMLoginProfile** 来创建带有密码的登录配置文件。如果 Bob 需要运行 Amazon PowerShell 或跨平台 CLI 命令或者 Amazon 进行 API 调用，则必须单独运行 **New-IAMAccessKey** 命令来创建访问密钥。

```
New-IAMUser -UserName Bob
```

输出：

```
Arn               : arn:aws:iam::123456789012:user/Bob
CreateDate        : 4/22/2015 12:02:11 PM
PasswordLastUsed  : 1/1/0001 12:00:00 AM
Path              : /
UserId            : AIDAJWGEFDMEMEXAMPLE1
UserName          : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 CreateUser。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.
```

```
    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 CreateUser。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Creates a user and their login profile
 #
 # @param user_name [String] The name of the user
 # @param initial_password [String] The initial password for the user
 # @return [String, nil] The ID of the user if created, or nil if an error
occurred
 def create_user(user_name, initial_password)
   response = @iam_client.create_user(user_name: user_name)
   @iam_client.wait_until(:user_exists, user_name: user_name)
   @iam_client.create_login_profile(
     user_name: user_name,
     password: initial_password,
     password_reset_required: true
```

```
    )
    @logger.info("User '#{user_name}' created successfully.")
    response.user.user_id
  rescue Aws::IAM::Errors::EntityAlreadyExists
    @logger.error("Error creating user '#{user_name}': user already exists.")
    nil
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating user '#{user_name}': #{e.message}")
    nil
  end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 CreateUser。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn create_user(client: &iamClient, user_name: &str) -> Result<User,
 iamError> {
    let response = client.create_user().user_name(user_name).send().await?;

    Ok(response.user.unwrap())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 CreateUser。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->createuser(
      iv_username = iv_user_name ).
    MESSAGE 'User created successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamentityalrdyexex.
    MESSAGE 'User already exists.' TYPE 'E'.
  CATCH /aws1/cx_iamlimitexceededex.
    MESSAGE 'Limit exceeded for IAM users.' TYPE 'E'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Entity does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 CreateUser。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3
```

```
    public func createUser(name: String) async throws -> String {
        let input = CreateUserInput(
            userName: name
        )
        do {
            let output = try await client.createUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
            }
            guard let id = user.userId else {
                throw ServiceHandlerError.noSuchUser
            }
            return id
        } catch {
            print("ERROR: createUser:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 CreateUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **CreateVirtualMfaDevice** 与 CLI 配合使用

以下代码示例演示如何使用 CreateVirtualMfaDevice。

CLI

Amazon CLI

创建虚拟 MFA 设备

此示例创建名为 BobsMFADevice 的新虚拟 MFA 设备。它会创建一个包含引导信息的、名为 QRCode.png 的文件，并将其放在 C:/ 目录中。本示例中使用的引导方法为 QRCodePNG。

```
aws iam create-virtual-mfa-device \
    --virtual-mfa-device-name BobsMFADevice \
    --outfile C:/QRCode.png \
    --bootstrap-method QRCodePNG
```

输出：

```
{
    "VirtualMFADevice": {
        "SerialNumber": "arn:aws:iam::210987654321:mfa/BobsMFADevice"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 中使用多重身份验证（MFA）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 CreateVirtualMfaDevice。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个新的虚拟 MFA 设备。第 2 行和第 3 行提取虚拟 MFA 软件程序创建账户所需的 **Base32StringSeed** 值（作为二维码的替代方案）。使用该值配置程序后，从程序中获取两个连续的身份验证码。最后，使用最后一条命令将虚拟 MFA 设备关联到 IAM 用户 **Bob**，并将账户与两个身份验证码同步。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

输出：

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

示例 2：此示例创建一个新虚拟 MFA 设备。第 2 行和第 3 行提取 **QRCodePNG** 值并将其写入文件。虚拟的 MFA 软件程序可以扫描此图像以创建账户（作为手动输入 Base32StringSeed 值的替代方法）。在虚拟 MFA 程序中创建账户后，获取两个连续的身份验证码，然后在最后一条命令中输入验证码，将虚拟 MFA 设备关联到 IAM 用户 **Bob** 并同步账户。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
```

```
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

输出：

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 CreateVirtualMfaDevice。

Tools for PowerShell V5

示例 1：此示例创建一个新的虚拟 MFA 设备。第 2 行和第 3 行提取虚拟 MFA 软件程序创建账户所需的 **Base32StringSeed** 值（作为二维码的替代方案）。使用该值配置程序后，从程序中获取两个连续的身份验证码。最后，使用最后一条命令将虚拟 MFA 设备关联到 IAM 用户 **Bob**，并将账户与两个身份验证码同步。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
$SR = New-Object System.IO.StreamReader($Device.Base32StringSeed)
$base32stringseed = $SR.ReadToEnd()
$base32stringseed
CZWZMCQNW4DEXAMPLE3VOUGXJFZYSUW7EXAMPLECR4NJFD65GX2SLUDW2EXAMPLE
```

输出：

```
-- Pause here to enter base-32 string seed code into virtual MFA program to
 register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

示例 2：此示例创建一个新虚拟 MFA 设备。第 2 行和第 3 行提取 **QRCodePNG** 值并将其写入文件。虚拟的 MFA 软件程序可以扫描此图像以创建账户（作为手动输入 Base32StringSeed 值的替代方法）。在虚拟 MFA 程序中创建账户后，获取两个连续的身份验证码，然后在最后一条命令中输入验证码，将虚拟 MFA 设备关联到 IAM 用户 **Bob** 并同步账户。

```
$Device = New-IAMVirtualMFADevice -VirtualMFADeviceName BobsMFADevice
```

```
$BR = New-Object System.IO.BinaryReader($Device.QRCodePNG)
$BR.ReadBytes($BR.BaseStream.Length) | Set-Content -Encoding Byte -Path
 QRCode.png
```

输出：

```
 -- Pause here to scan PNG with virtual MFA program to register account. --

Enable-IAMMFADevice -SerialNumber $Device.SerialNumber -UserName Bob -
AuthenticationCode1 123456 -AuthenticationCode2 789012
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 CreateVirtualMfaDevice。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeactivateMfaDevice** 与 CLI 配合使用

以下代码示例演示如何使用 DeactivateMfaDevice。

CLI

Amazon CLI

停用 MFA 设备

此命令使用与用户 Bob 关联的 ARN arn:aws:iam::210987654321:mfa/BobsMFADevice 停用虚拟 MFA 设备。

```
aws iam deactivate-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 中使用多重身份验证（MFA）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeactivateMfaDevice。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令禁用与具有序列号 **123456789012** 的用户 **Bob** 关联的硬件 MFA 设备。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

示例 2：此命令禁用与具有 ARN **arn:aws:iam::210987654321:mfa/David** 的用户 **David** 关联的虚拟 MFA 设备。请注意，虚拟 MFA 设备不会从账户中删除。虚拟设备仍然存在并出现在 **Get-IAMVirtualMFADevice** 命令的输出中。您必须先使用 **Remove-IAMVirtualMFADevice** 命令删除旧的虚拟 MFA 设备，然后才能为同一用户创建新的虚拟 MFA 设备。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeactivateMfaDevice。

Tools for PowerShell V5

示例 1：此命令禁用与具有序列号 **123456789012** 的用户 **Bob** 关联的硬件 MFA 设备。

```
Disable-IAMMFADevice -UserName "Bob" -SerialNumber "123456789012"
```

示例 2：此命令禁用与具有 ARN **arn:aws:iam::210987654321:mfa/David** 的用户 **David** 关联的虚拟 MFA 设备。请注意，虚拟 MFA 设备不会从账户中删除。虚拟设备仍然存在并出现在 **Get-IAMVirtualMFADevice** 命令的输出中。您必须先使用 **Remove-IAMVirtualMFADevice** 命令删除旧的虚拟 MFA 设备，然后才能为同一用户创建新的虚拟 MFA 设备。

```
Disable-IAMMFADevice -UserName "David" -SerialNumber
 "arn:aws:iam::210987654321:mfa/David"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeactivateMfaDevice。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteAccessKey** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteAccessKey。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户
- 管理访问密钥

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Delete an IAM user's access key.
    /// </summary>
    /// <param name="accessKeyId">The Id for the IAM access key.</param>
    /// <param name="userName">The username of the user that owns the IAM
    /// access key.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteAccessKeyAsync(string accessKeyId, string
userName)
    {
        var response = await _IAMService.DeleteAccessKeyAsync(new
DeleteAccessKeyRequest
        {
            AccessKeyId = accessKeyId,
            UserName = userName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteAccessKey。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_access_key
#
# This function deletes an IAM access key for the specified IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to delete.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_access_key() {
  local user_name access_key response
  local option OPTARG # Required to use getopts command in a function.
```

```
 # bashsupport disable=BP5008
 function usage() {
   echo "function iam_delete_access_key"
   echo "Deletes an AWS Identity and Access Management (IAM) access key for the
specified IAM user"
   echo "  -u user_name    The name of the user."
   echo "  -k access_key   The access key to delete."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "u:k:h" option; do
   case "${option}" in
     u) user_name="${OPTARG}" ;;
     k) access_key="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$user_name" ]]; then
   errecho "ERROR: You must provide a username with the -u parameter."
   usage
   return 1
 fi

 if [[ -z "$access_key" ]]; then
   errecho "ERROR: You must provide an access key with the -k parameter."
   usage
   return 1
 fi

 iecho "Parameters:\n"
 iecho "    Username:   $user_name"
 iecho "    Access key:   $access_key"
 iecho ""
```

```
  response=$(aws iam delete-access-key \
    --user-name "$user_name" \
    --access-key-id "$access_key")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-access-key operation failed.\n$response"
    return 1
  fi

  iecho "delete-access-key response:$response"
  iecho

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 DeleteAccessKey。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::deleteAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
                                  const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
```

```
    auto outcome = iam.DeleteAccessKey(request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting access key " << accessKeyID << " from user "
                  << userName << ": " << outcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
        std::cout << "Successfully deleted access key " << accessKeyID
                  << " for IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DeleteAccessKey。

CLI

Amazon CLI

删除 IAM 用户的访问密钥

以下 `delete-access-key` 命令将删除名为 Bob IAM 用户的指定访问密钥（访问密钥 ID 和秘密访问密钥）。

```
aws iam delete-access-key \
    --access-key-id AKIDPMS9RO4H3FEXAMPLE \
    --user-name Bob
```

此命令不生成任何输出。

要列出为 IAM 用户定义的访问密钥，请使用 `list-access-keys` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的访问密钥。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteAccessKey。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteAccessKey deletes an access key from a user.
func (wrapper UserWrapper) DeleteAccessKey(ctx context.Context, userName string,
 keyId string) error {
 _, err := wrapper.IamClient.DeleteAccessKey(ctx, &iam.DeleteAccessKeyInput{
  AccessKeyId: aws.String(keyId),
  UserName:    aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete access key %v. Here's why: %v\n", keyId, err)
 }
 return err
```

```
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeleteAccessKey。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteAccessKeyRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccessKey {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessKey>\s

                Where:
                    username - The name of the user.\s
```

```
                    accessKey - The access key ID for the secret access key you
 want to delete.\s
                """";

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessKey = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        deleteKey(iam, username, accessKey);
        iam.close();
    }

    public static void deleteKey(IamClient iam, String username, String
 accessKey) {
        try {
            DeleteAccessKeyRequest request = DeleteAccessKeyRequest.builder()
                    .accessKeyId(accessKey)
                    .userName(username)
                    .build();

            iam.deleteAccessKey(request);
            System.out.println("Successfully deleted access key " + accessKey +
                    " from user " + username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 DeleteAccessKey。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除访问密钥。

```
import { DeleteAccessKeyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const deleteAccessKey = (userName, accessKeyId) => {
  const command = new DeleteAccessKeyCommand({
    AccessKeyId: accessKeyId,
    UserName: userName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteAccessKey。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  UserName: "USER_NAME",
};

iam.deleteAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteAccessKey。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun deleteKey(
    userNameVal: String,
    accessKey: String,
) {
    val request =
        DeleteAccessKeyRequest {
            accessKeyId = accessKey
            userName = userNameVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteAccessKey(request)
        println("Successfully deleted access key $accessKey from $userNameVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 DeleteAccessKey。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从名为 **Bob** 的用户删除密钥 ID 为 **AKIAIOSFODNN7EXAMPLE** 的 Amazon 访问密钥对。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteAccessKey。

Tools for PowerShell V5

示例 1：此示例从名为 **Bob** 的用户删除密钥 ID 为 **AKIAIOSFODNN7EXAMPLE** 的 Amazon 访问密钥对。

```
Remove-IAMAccessKey -AccessKeyId AKIAIOSFODNN7EXAMPLE -UserName Bob -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteAccessKey。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 DeleteAccessKey。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end

  # Lists access keys for a user
  #
  # @param user_name [String] The name of the user.
  def list_access_keys(user_name)
    response = @iam_client.list_access_keys(user_name: user_name)
    if response.access_key_metadata.empty?
      @logger.info("No access keys found for user '#{user_name}'.")
    else
      response.access_key_metadata.map(&:access_key_id)
    end
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
    []
  rescue StandardError => e
    @logger.error("Error listing access keys: #{e.message}")
    []
  end

  # Creates an access key for a user
  #
  # @param user_name [String] The name of the user.
  # @return [Boolean]
  def create_access_key(user_name)
```

```
    response = @iam_client.create_access_key(user_name: user_name)
    access_key = response.access_key
    @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
    access_key
  rescue Aws::IAM::Errors::LimitExceeded
    @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
    nil
  rescue StandardError => e
    @logger.error("Error creating access key: #{e.message}")
    nil
  end

  # Deactivates an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def deactivate_access_key(user_name, access_key_id)
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
```

```
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteAccessKey。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_access_key(
    client: &iamClient,
    user: &User,
    key: &AccessKey,
) -> Result<(), iamError> {
    loop {
        match client
            .delete_access_key()
            .user_name(user.user_name())
            .access_key_id(key.access_key_id())
            .send()
            .await
        {
            Ok(_) => {
                break;
            }
            Err(e) => {
                println!("Can't delete the access key: {:?}", e);
                sleep(Duration::from_secs(2)).await;
            }
        }
    }
    Ok(())
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DeleteAccessKey。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    lo_iam->deleteaccesskey(
      iv_accesskeyid = iv_access_key_id
      iv_username = iv_user_name ).
    MESSAGE 'Access key deleted successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key or user does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeleteAccessKey。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func deleteAccessKey(user: IAMClientTypes.User? = nil,
                                key: IAMClientTypes.AccessKey) async throws
    {
        let userName: String?

        if user != nil {
            userName = user!.userName
        } else {
            userName = nil
        }

        let input = DeleteAccessKeyInput(
            accessKeyId: key.accessKeyId,
            userName: userName
        )
        do {
            _ = try await iamClient.deleteAccessKey(input: input)
        } catch {
            print("ERROR: deleteAccessKey:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DeleteAccessKey。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteAccountAlias** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteAccountAlias。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::deleteAccountAlias(const Aws::String &accountAlias,
                                     const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteAccountAliasRequest request;
    request.SetAccountAlias(accountAlias);

    const auto outcome = iam.DeleteAccountAlias(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting account alias " << accountAlias << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted account alias " << accountAlias <<
                  std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DeleteAccountAlias。

CLI

Amazon CLI

删除账户别名

以下 delete-account-alias 命令将删除当前账户的别名 mycompany。

```
aws iam delete-account-alias \
    --account-alias mycompany
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的您的 Amazon 账户 ID 及其别名。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteAccountAlias。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.DeleteAccountAliasRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteAccountAlias {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <alias>\s
```

```
                Where:
                    alias - The account alias to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String alias = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMAccountAlias(iam, alias);
        iam.close();
    }

    public static void deleteIAMAccountAlias(IamClient iam, String alias) {
        try {
            DeleteAccountAliasRequest request =
 DeleteAccountAliasRequest.builder()
                    .accountAlias(alias)
                    .build();

            iam.deleteAccountAlias(request);
            System.out.println("Successfully deleted account alias " + alias);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 DeleteAccountAlias。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除账户别名。

```
import { DeleteAccountAliasCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} alias
 */
export const deleteAccountAlias = (alias) => {
  const command = new DeleteAccountAliasCommand({ AccountAlias: alias });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteAccountAlias。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteAccountAlias({ AccountAlias: process.argv[2] }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteAccountAlias。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun deleteIAMAccountAlias(alias: String) {
    val request =
        DeleteAccountAliasRequest {
            accountAlias = alias
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
```

```
        iamClient.deleteAccountAlias(request)
        println("Successfully deleted account alias $alias")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 DeleteAccountAlias。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从您的 Amazon Web Services 账户 中删除账户别名。使用别名的用户登录页面 https://mycompanyaws.signin.aws.amazon.com/console 不再有效。您必须将原始 URL 改为使用您的 Amazon Web Services 账户 ID 号：https://<accountidnumber>.signin.aws.amazon.com/console。

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteAccountAlias。

Tools for PowerShell V5

示例 1：此示例从您的 Amazon Web Services 账户 中删除账户别名。使用别名的用户登录页面 https://mycompanyaws.signin.aws.amazon.com/console 不再有效。您必须将原始 URL 改为使用您的 Amazon Web Services 账户 ID 号：https://<accountidnumber>.signin.aws.amazon.com/console。

```
Remove-IAMAccountAlias -AccountAlias mycompanyaws
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteAccountAlias。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 DeleteAccountAlias。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end

  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteAccountAlias。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->deleteaccountalias(
          iv_accountalias = iv_account_alias ).
        MESSAGE 'Account alias deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Account alias does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeleteAccountAlias。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteAccountPasswordPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteAccountPasswordPolicy。

CLI

Amazon CLI

删除当前账户密码策略

以下 `delete-account-password-policy` 命令将删除当前账户的密码策略。

```
aws iam delete-account-password-policy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的为 IAM 用户设置账户密码策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteAccountPasswordPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除 Amazon Web Services 账户 的密码策略并将所有值重置为其原始默认值。如果当前不存在密码策略，则会显示以下错误消息：找不到名为 PasswordPolicy 的账户策略。

```
Remove-IAMAccountPasswordPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteAccountPasswordPolicy。

Tools for PowerShell V5

示例 1：此示例删除 Amazon Web Services 账户 的密码策略并将所有值重置为其原始默认值。如果当前不存在密码策略，则会显示以下错误消息：找不到名为 PasswordPolicy 的账户策略。

```
Remove-IAMAccountPasswordPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteAccountPasswordPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteGroup** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteGroup。

CLI

Amazon CLI

删除 IAM 组

以下 delete-group 命令将删除名为 MyTestGroup 的 IAM 组。

```
aws iam delete-group \
    --group-name MyTestGroup
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的删除 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteGroup。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { DeleteGroupCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} groupName
 */
export const deleteGroup = async (groupName) => {
  const command = new DeleteGroupCommand({
```

```
    GroupName: groupName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyTestGroup** 的 IAM 组。第一条命令删除作为该组成员的所有 IAM 用户，第二条命令删除该 IAM 组。这两条命令都可以在没有任何确认提示的情况下生效。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteGroup。

Tools for PowerShell V5

示例 1：此示例删除名为 **MyTestGroup** 的 IAM 组。第一条命令删除作为该组成员的所有 IAM 用户，第二条命令删除该 IAM 组。这两条命令都可以在没有任何确认提示的情况下生效。

```
(Get-IAMGroup -GroupName MyTestGroup).Users | Remove-IAMUserFromGroup -GroupName
 MyTestGroup -Force
Remove-IAMGroup -GroupName MyTestGroup -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteGroupPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteGroupPolicy。

CLI

Amazon CLI

从 IAM 组中删除策略

以下 delete-group-policy 命令可将名为 ExamplePolicy 的策略从名为 Admins 的组中删除。

```
aws iam delete-group-policy \
    --group-name Admins \
    --policy-name ExamplePolicy
```

此命令不生成任何输出。

要查看附加到组的策略，请使用 list-group-policies 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM policy。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteGroupPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从 IAM 组 **Testers** 中删除名为 **TesterPolicy** 的内联策略。该组中的用户会立即失去该策略中定义的权限。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteGroupPolicy。

Tools for PowerShell V5

示例 1：此示例从 IAM 组 **Testers** 中删除名为 **TesterPolicy** 的内联策略。该组中的用户会立即失去该策略中定义的权限。

```
Remove-IAMGroupPolicy -GroupName Testers -PolicyName TestPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteGroupPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteInstanceProfile** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteInstanceProfile。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 构建和管理弹性服务
- Amazon MSK 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
```

```
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
        }
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteInstanceProfile。

CLI

Amazon CLI

删除实例配置文件

以下 delete-instance-profile 命令将删除名为 ExampleInstanceProfile 的实例配置文件。

```
aws iam delete-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteInstanceProfile。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const client = new IAMClient({});
await client.send(
  new DeleteInstanceProfileCommand({
    InstanceProfileName: NAMES.instanceProfileName,
  }),
);
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteInstanceProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyAppInstanceProfile** 的 EC2 实例配置文件。第一条命令从实例配置文件中分离所有角色，然后第二条命令删除实例配置文件。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteInstanceProfile。

Tools for PowerShell V5

示例 1：此示例删除名为 **MyAppInstanceProfile** 的 EC2 实例配置文件。第一条命令从实例配置文件中分离所有角色，然后第二条命令删除实例配置文件。

```
(Get-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile).Roles |
 Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyAppInstanceProfile
Remove-IAMInstanceProfile -InstanceProfileName MyAppInstanceProfile
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteInstanceProfile。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例从实例配置文件中移除该角色，分离附加到该角色的所有策略，然后删除所有资源。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
```

```
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
```

```
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )
```

- 有关 API 详细信息，请参阅《适用于 Python（Boto3）的 Amazon SDK API 参考》中的 DeleteInstanceProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteLoginProfile** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteLoginProfile。

CLI

Amazon CLI

删除 IAM 用户的密码

以下 delete-login-profile 命令删除名为 Bob 的 IAM 用户的密码。

```
aws iam delete-login-profile \
    --user-name Bob
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的密码。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteLoginProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **Bob** 的 IAM 用户的登录配置文件。这可以阻止用户登录 Amazon 控制台。它不会阻止用户使用可能仍附加到用户账户的 Amazon 访问密钥运行任何 Amazon CLI、PowerShell 或 API 调用。

```
Remove-IAMLoginProfile -UserName Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteLoginProfile。

Tools for PowerShell V5

示例 1：此示例删除名为 **Bob** 的 IAM 用户的登录配置文件。这可以阻止用户登录 Amazon 控制台。它不会阻止用户使用可能仍附加到用户账户的 Amazon 访问密钥运行任何 Amazon CLI、PowerShell 或 API 调用。

```
Remove-IAMLoginProfile -UserName Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteLoginProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteOpenIdConnectProvider** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteOpenIdConnectProvider。

CLI

Amazon CLI

删除 IAM OpenID Connect 身份提供者

此示例删除了连接到提供者 example.oidcprovider.com 的 IAM OIDC 提供者。

```
aws iam delete-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteOpenIdConnectProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除了连接到提供者 **example.oidcprovider.com** 的 IAM OIDC 提供者。确保更新或删除角色信任策略 **Principal** 元素中引用此提供者的所有角色。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteOpenIdConnectProvider。

Tools for PowerShell V5

示例 1：此示例删除了连接到提供者 **example.oidcprovider.com** 的 IAM OIDC 提供者。确保更新或删除角色信任策略 **Principal** 元素中引用此提供者的所有角色。

```
Remove-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteOpenIdConnectProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeletePolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeletePolicy`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户
- 创建 Amazon Managed Grafana 工作区
- Amazon MSK 入门
- Step Functions 入门
- 管理策略

- 设置 Systems Manager

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Delete an IAM policy.
    /// </summary>
    /// <param name="policyArn">The Amazon Resource Name (ARN) of the policy to
    /// delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeletePolicyAsync(string policyArn)
    {
        var response = await _IAMService.DeletePolicyAsync(new
DeletePolicyRequest { PolicyArn = policyArn });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeletePolicy。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_policy
#
# This function deletes an IAM policy.
#
# Parameters:
#       -n policy_arn -- The name of the IAM policy arn.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_delete_policy() {
  local policy_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_policy"
    echo "Deletes an AWS Identity and Access Management (IAM) policy"
    echo "  -n policy_arn -- The name of the IAM policy arn."
    echo ""
```

```
}

# Retrieve the calling parameters.
while getopts "n:h" option; do
  case "${option}" in
    n) policy_arn="${OPTARG}" ;;
    h)
      usage
      return 0
      ;;
    \?)
      echo "Invalid parameter"
      usage
      return 1
      ;;
  esac
done
export OPTIND=1

if [[ -z "$policy_arn" ]]; then
  errecho "ERROR: You must provide a policy arn with the -n parameter."
  usage
  return 1
fi

iecho "Parameters:\n"
iecho "    Policy arn:  $policy_arn"
iecho ""

response=$(aws iam delete-policy \
  --policy-arn "$policy_arn")

local error_code=${?}

if [[ $error_code -ne 0 ]]; then
  aws_cli_error_log $error_code
  errecho "ERROR: AWS reports delete-policy operation failed.\n$response"
  return 1
fi

iecho "delete-policy response:$response"
iecho

return 0
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 DeletePolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::deletePolicy(const Aws::String &policyArn,
                               const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeletePolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.DeletePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting policy with arn " << policyArn << ": "
                  << outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully deleted policy with arn " << policyArn
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DeletePolicy。

CLI

Amazon CLI

删除 IAM 策略

此示例删除了 ARN 为 arn:aws:iam::123456789012:policy/MySamplePolicy 的策略。

```
aws iam delete-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeletePolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
```

```
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// DeletePolicy deletes a policy.
func (wrapper PolicyWrapper) DeletePolicy(ctx context.Context, policyArn string)
 error {
 _, err := wrapper.IamClient.DeletePolicy(ctx, &iam.DeletePolicyInput{
  PolicyArn: aws.String(policyArn),
 })
 if err != nil {
  log.Printf("Couldn't delete policy %v. Here's why: %v\n", policyArn, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeletePolicy。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.DeletePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
```

```
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeletePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <policyARN>\s

                Where:
                    policyARN - A policy ARN value to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String policyARN = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMPolicy(iam, policyARN);
        iam.close();
    }

    public static void deleteIAMPolicy(IamClient iam, String policyARN) {
        try {
            DeletePolicyRequest request = DeletePolicyRequest.builder()
                    .policyArn(policyARN)
                    .build();

            iam.deletePolicy(request);
            System.out.println("Successfully deleted the policy");

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
```

```
            System.exit(1);
        }
        System.out.println("Done");
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 DeletePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除策略。

```
import { DeletePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const deletePolicy = (policyArn) => {
  const command = new DeletePolicyCommand({ PolicyArn: policyArn });
  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeletePolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun deleteIAMPolicy(policyARNVal: String?) {
    val request =
        DeletePolicyRequest {
            policyArn = policyARNVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deletePolicy(request)
        println("Successfully deleted $policyARNVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 DeletePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略。必须先通过运行 **Remove-IAMPolicyVersion** 删除默认版本除外的所有版本，然后才能删除该策略。您还必须将该策略与所有 IAM 用户、组或角色分离。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

示例 2：此示例通过先删除所有非默认策略版本，将其与所有附加的 IAM 实体分离，最后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后删除集合中的每个策略。第三行检索该策略附加到的所有 IAM 用户、组和角色。第四行到第六行将该策略与每个附加的实体分离。最后一行使用此命令删除托管策略，以及剩余的默认版本。该示例在需要抑制确认提示的所有行中都包含 **-Force** 开关参数。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeletePolicy。

Tools for PowerShell V5

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略。必须先通过运行 **Remove-IAMPolicyVersion** 删除默认版本除外的所有版本，然后才能删除该策略。您还必须将该策略与所有 IAM 用户、组或角色分离。

```
Remove-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

示例 2：此示例通过先删除所有非默认策略版本，将其与所有附加的 IAM 实体分离，最后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后删除集合中的每个策略。第三行检索该策略附加到的所有 IAM 用户、组和角色。第四行到第六行将该策略与每个附加的实体分离。最后一行使用此命令删除托管策略，以及剩余的默认版本。该示例在需要抑制确认提示的所有行中都包含 **-Force** 开关参数。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
$attached = Get-IAMEntitiesForPolicy -PolicyArn $pol.Arn
$attached.PolicyGroups | Unregister-IAMGroupPolicy -PolicyArn $pol.arn
$attached.PolicyRoles | Unregister-IAMRolePolicy -PolicyArn $pol.arn
$attached.PolicyUsers | Unregister-IAMUserPolicy -PolicyArn $pol.arn
Remove-IAMPolicy $pol.Arn -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeletePolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 DeletePolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_policy(client: &iamClient, policy: Policy) -> Result<(),
 iamError> {
    client
        .delete_policy()
        .policy_arn(policy.arn.unwrap())
        .send()
        .await?;
    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DeletePolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->deletepolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Policy cannot be deleted due to attachments.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeletePolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func deletePolicy(policy: IAMClientTypes.Policy) async throws {
        let input = DeletePolicyInput(
            policyArn: policy.arn
        )
        do {
            _ = try await iamClient.deletePolicy(input: input)
        } catch {
            print("ERROR: deletePolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DeletePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeletePolicyVersion** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeletePolicyVersion。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 管理策略

- 回滚策略版本

CLI

Amazon CLI

删除托管策略的某个版本

此示例从 ARN 为 arn:aws:iam::123456789012:policy/MySamplePolicy 的策略中删除标识为 v2 的版本。

```
aws iam delete-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeletePolicyVersion。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略中删除标识为 **v2** 的版本。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

示例 2：此示例通过先删除所有非默认策略版本，然后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后使用此命令删除集合中的每个策略。最后一行删除策略本身，以及剩余的默认版本。请注意，要成功删除托管策略，还必须使用 **Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy** 和 **Unregister-IAMRolePolicy** 命令将该策略与所有用户、组或角色分离。请参阅 **Remove-IAMPolicy** cmdlet 的示例。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
```

```
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeletePolicyVersion。

Tools for PowerShell V5

示例 1：此示例从 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的策略中删除标识为 **v2** 的版本。

```
Remove-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MySamplePolicy -VersionID v2
```

示例 2：此示例通过先删除所有非默认策略版本，然后删除策略本身来删除策略。第一行检索策略对象。第二行检索集合中未标记为默认的所有策略版本，然后使用此命令删除集合中的每个策略。最后一行删除策略本身，以及剩余的默认版本。请注意，要成功删除托管策略，还必须使用 **Unregister-IAMUserPolicy**、**Unregister-IAMGroupPolicy** 和 **Unregister-IAMRolePolicy** 命令将该策略与所有用户、组或角色分离。请参阅 **Remove-IAMPolicy** cmdlet 的示例。

```
$pol = Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
Get-IAMPolicyVersions -PolicyArn $pol.Arn | where {-not $_.IsDefaultVersion} |
 Remove-IAMPolicyVersion -PolicyArn $pol.Arn -force
Remove-IAMPolicy -PolicyArn $pol.Arn -force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeletePolicyVersion。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
```

```
        lo_iam->deletepolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Policy version deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Cannot delete default policy version.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeletePolicyVersion。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteRole** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteRole。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 配置 Amazon ECS Service Connect
- 使用 Lambda 代理集成创建 REST API
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 Amazon Managed Grafana 工作区
- 创建第一个 Lambda 函数
- Redshift Serverless 入门
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon Redshift 预置集群入门
- Amazon SageMaker Feature Store 入门

- Config 入门
- IoT Device Defender 入门
- Step Functions 入门
- 管理角色
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Delete an IAM role.
    /// </summary>
    /// <param name="roleName">The name of the IAM role to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRoleAsync(string roleName)
    {
        var response = await _IAMService.DeleteRoleAsync(new DeleteRoleRequest
{ RoleName = roleName });
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteRole。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_role
#
# This function deletes an IAM role.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
```

```
################################################################################
function iam_delete_role() {
  local role_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_role"
    echo "Deletes an AWS Identity and Access Management (IAM) role"
    echo "  -n role_name -- The name of the IAM role."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "n:h" option; do
    case "${option}" in
      n) role_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  echo "role_name:$role_name"
  if [[ -z "$role_name" ]]; then
    errecho "ERROR: You must provide a role name with the -n parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
  iecho "    Role name:  $role_name"
  iecho ""

  response=$(aws iam delete-role \
    --role-name "$role_name")
```

```
  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-role operation failed.\n$response"
    return 1
  fi

  iecho "delete-role response:$response"
  iecho

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteRole。

CLI

Amazon CLI

删除 IAM 角色

以下 delete-role 命令将删除名为 Test-Role 的角色。

```
aws iam delete-role \
    --role-name Test-Role
```

此命令不生成任何输出。

在删除角色之前，必须从所有实例配置文件中删除该角色（remove-role-from-instance-profile），分离所有托管策略（detach-role-policy），删除附加到该角色的所有内联策略（delete-role-policy）。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色和使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteRole。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DeleteRole deletes a role. All attached policies must be detached before a
// role can be deleted.
func (wrapper RoleWrapper) DeleteRole(ctx context.Context, roleName string) error
 {
 _, err := wrapper.IamClient.DeleteRole(ctx, &iam.DeleteRoleInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't delete role %v. Here's why: %v\n", roleName, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeleteRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除角色。

```
import { DeleteRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const deleteRole = (roleName) => {
  const command = new DeleteRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从当前 IAM 账户中删除名为 **MyNewRole** 的角色。必须先使用 **Unregister-IAMRolePolicy** 命令分离所有托管策略，然后才能删除该角色。内联策略将与角色一起删除。

```
Remove-IAMRole -RoleName MyNewRole
```

示例 2：此示例将所有托管策略与名为 **MyNewRole** 的角色分离，然后删除该角色。第一行检索作为集合附加到该角色的所有托管策略，然后将集合中的每个策略与该角色分离。第二行删除该角色本身。内联策略将与角色一起删除。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteRole。

Tools for PowerShell V5

示例 1：此示例从当前 IAM 账户中删除名为 **MyNewRole** 的角色。必须先使用 **Unregister-IAMRolePolicy** 命令分离所有托管策略，然后才能删除该角色。内联策略将与角色一起删除。

```
Remove-IAMRole -RoleName MyNewRole
```

示例 2：此示例将所有托管策略与名为 **MyNewRole** 的角色分离，然后删除该角色。第一行检索作为集合附加到该角色的所有托管策略，然后将集合中的每个策略与该角色分离。第二行删除该角色本身。内联策略将与角色一起删除。

```
Get-IAMAttachedRolePolicyList -RoleName MyNewRole | Unregister-IAMRolePolicy -
RoleName MyNewRole
Remove-IAMRole -RoleName MyNewRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 DeleteRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
  # Deletes a role and its attached policies.
```

```
  #
  # @param role_name [String] The name of the role to delete.
  def delete_role(role_name)
    # Detach and delete attached policies
    @iam_client.list_attached_role_policies(role_name: role_name).each do |
response|
      response.attached_policies.each do |policy|
        @iam_client.detach_role_policy({
                                         role_name: role_name,
                                         policy_arn: policy.policy_arn
                                       })
        # Check if the policy is a customer managed policy (not AWS managed)
        unless policy.policy_arn.include?('aws:policy/')
          @iam_client.delete_policy({ policy_arn: policy.policy_arn })
          @logger.info("Deleted customer managed policy #{policy.policy_name}.")
        end
      end
    end

    # Delete the role
    @iam_client.delete_role({ role_name: role_name })
    @logger.info("Deleted role #{role_name}.")
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't detach policies and delete role #{role_name}. Here's
 why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_role(client: &iamClient, role: &Role) -> Result<(), iamError>
 {
    let role = role.clone();
    while client
        .delete_role()
        .role_name(role.role_name())
        .send()
        .await
        .is_err()
    {
        sleep(Duration::from_secs(2)).await;
    }
    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DeleteRole。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->deleterole( iv_rolename = iv_role_name ).
        MESSAGE 'Role deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'Role cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeleteRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func deleteRole(role: IAMClientTypes.Role) async throws {
        let input = DeleteRoleInput(
            roleName: role.roleName
        )
        do {
            _ = try await iamClient.deleteRole(input: input)
        } catch {
            print("ERROR: deleteRole:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DeleteRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`DeleteRolePermissionsBoundary`** 与 CLI 配合使用

以下代码示例演示如何使用 `DeleteRolePermissionsBoundary`。

CLI

Amazon CLI

删除 IAM 角色的权限边界

以下 delete-role-permissions-boundary 示例删除指定 IAM 角色的权限边界。要对角色应用权限边界，请使用 put-role-permissions-boundary 命令。

```
aws iam delete-role-permissions-boundary \
    --role-name lambda-application-role
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteRolePermissionsBoundary。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例展示了如何删除附加到 IAM 角色的权限边界。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteRolePermissionsBoundary。

Tools for PowerShell V5

示例 1：此示例展示了如何删除附加到 IAM 角色的权限边界。

```
Remove-IAMRolePermissionsBoundary -RoleName MyRoleName
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteRolePermissionsBoundary。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteRolePolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteRolePolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- Redshift Serverless 入门
- Amazon Redshift 预置集群入门
- Config 入门
- IoT Device Defender 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Delete an IAM role policy.
    /// </summary>
    /// <param name="roleName">The name of the IAM role.</param>
    /// <param name="policyName">The name of the IAM role policy to delete.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteRolePolicyAsync(string roleName, string
 policyName)
    {
        var response = await _IAMService.DeleteRolePolicyAsync(new
 DeleteRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
        });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteRolePolicy。

CLI

Amazon CLI

从 IAM 角色中移除策略

以下 delete-role-policy 命令可将名为 ExamplePolicy 的策略从名为 Test-Role 的角色中移除。

```
aws iam delete-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteRolePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { DeleteRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
```

```
 * @param {string} roleName
 * @param {string} policyName
 */
export const deleteRolePolicy = (roleName, policyName) => {
  const command = new DeleteRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
  });
  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteRolePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除了 IAM 角色 **S3BackupRole** 中嵌入的内联策略 **S3AccessPolicy**。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteRolePolicy。

Tools for PowerShell V5

示例 1：此示例删除了 IAM 角色 **S3BackupRole** 中嵌入的内联策略 **S3AccessPolicy**。

```
Remove-IAMRolePolicy -PolicyName S3AccessPolicy -RoleName S3BackupRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteSAMLProvider** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteSAMLProvider。

CLI

Amazon CLI

删除 SAML 提供者

此示例删除了 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider 的 IAM SAML 2.0 提供者。

```
aws iam delete-saml-provider \
--saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM SAML 身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteSAMLProvider。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { DeleteSAMLProviderCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} providerArn
 * @returns
 */
export const deleteSAMLProvider = async (providerArn) => {
  const command = new DeleteSAMLProviderCommand({
    SAMLProviderArn: providerArn,
  });
```

```
  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteSAMLProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider** 的 IAM SAML 2.0 提供者。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteSAMLProvider。

Tools for PowerShell V5

示例 1：此示例删除了 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFSProvider** 的 IAM SAML 2.0 提供者。

```
Remove-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFSProvider
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteSAMLProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteServerCertificate** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteServerCertificate。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::deleteServerCertificate(const Aws::String &certificateName,
                                          const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::DeleteServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    const auto outcome = iam.DeleteServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error deleting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        std::cout << "Successfully deleted server certificate " <<
 certificateName
                  << std::endl;
    }

    return result;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DeleteServerCertificate。

CLI

Amazon CLI

从 Amazon 账户中删除服务器证书

以下 `delete-server-certificate` 命令可从 Amazon 账户中删除指定的服务器证书。

```
aws iam delete-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

此命令不生成任何输出。

要列出 Amazon 账户中可用的服务器证书，请使用 `list-server-certificates` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 中管理服务器证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteServerCertificate。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除服务器证书。

```
import { DeleteServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
```

```
 */
export const deleteServerCertificate = (certName) => {
  const command = new DeleteServerCertificateCommand({
    ServerCertificateName: certName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteServerCertificate。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.deleteServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
);
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-deleting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteServerCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **MyServerCert** 的服务器证书。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteServerCertificate。

Tools for PowerShell V5

示例 1：此示例删除名为 **MyServerCert** 的服务器证书。

```
Remove-IAMServerCertificate -ServerCertificateName MyServerCert
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteServerCertificate。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
```

```
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteServerCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteServiceLinkedRole** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteServiceLinkedRole。

CLI

Amazon CLI

删除服务相关角色

以下 delete-service-linked-role 示例将删除您不再需要的指定服务相关角色。删除操作异步进行。您也可以使用 get-service-linked-role-deletion-status 命令查看删除状态并确认何时删除。

```
aws iam delete-service-linked-role \
    --role-name AWSServiceRoleForLexBots
```

输出：

```
{
    "DeletionTaskId": "task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用服务相关角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteServiceLinkedRole。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
```

```
}



// DeleteServiceLinkedRole deletes a service-linked role.
func (wrapper RoleWrapper) DeleteServiceLinkedRole(ctx context.Context, roleName
 string) error {
 _, err := wrapper.IamClient.DeleteServiceLinkedRole(ctx,
 &iam.DeleteServiceLinkedRoleInput{
  RoleName: aws.String(roleName)},
 )
 if err != nil {
  log.Printf("Couldn't delete service-linked role %v. Here's why: %v\n",
 roleName, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeleteServiceLinkedRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { DeleteServiceLinkedRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
```

```
export const deleteServiceLinkedRole = (roleName) => {
  const command = new DeleteServiceLinkedRoleCommand({ RoleName: roleName });
  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteServiceLinkedRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除服务关联角色。请注意，如果服务仍在使用此角色，则此命令将导致失败。

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteServiceLinkedRole。

Tools for PowerShell V5

示例 1：此示例删除服务关联角色。请注意，如果服务仍在使用此角色，则此命令将导致失败。

```
Remove-IAMServiceLinkedRole -RoleName
 AWSServiceRoleForAutoScaling_RoleNameEndsWithThis
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteServiceLinkedRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Deletes a service-linked role.
#
# @param role_name [String] The name of the role to delete.
def delete_service_linked_role(role_name)
  response = @iam_client.delete_service_linked_role(role_name: role_name)
  task_id = response.deletion_task_id
  check_deletion_status(role_name, task_id)
rescue Aws::Errors::ServiceError => e
  handle_deletion_error(e, role_name)
end

private

# Checks the deletion status of a service-linked role
#
# @param role_name [String] The name of the role being deleted
# @param task_id [String] The task ID for the deletion process
def check_deletion_status(role_name, task_id)
  loop do
    response = @iam_client.get_service_linked_role_deletion_status(
      deletion_task_id: task_id
    )
    status = response.status
    @logger.info("Deletion of #{role_name} #{status}.")
    break if %w[SUCCEEDED FAILED].include?(status)

    sleep(3)
  end
end

# Handles deletion error
#
# @param e [Aws::Errors::ServiceError] The error encountered during deletion
# @param role_name [String] The name of the role attempted to delete
def handle_deletion_error(e, role_name)
  return if e.code == 'NoSuchEntity'

  @logger.error("Couldn't delete #{role_name}. Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  raise
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteServiceLinkedRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_service_linked_role(
    client: &iamClient,
    role_name: &str,
) -> Result<(), iamError> {
    client
        .delete_service_linked_role()
        .role_name(role_name)
        .send()
        .await?;

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API 参考》中的 DeleteServiceLinkedRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteSigningCertificate** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteSigningCertificate。

CLI

Amazon CLI

删除 IAM 用户的签名证书

以下 `delete-signing-certificate` 命令删除名为 Bob 的 IAM 用户的指定签名证书。

```
aws iam delete-signing-certificate \
    --user-name Bob \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE
```

此命令不生成任何输出。

要获取签名证书的 ID，请使用 `list-signing-certificates` 命令。

有关更多信息，请参阅《Amazon EC2 用户指南》中的管理签名证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteSigningCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从名为 **Bob** 的 IAM 用户删除 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的签名证书。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteSigningCertificate。

Tools for PowerShell V5

示例 1：此示例从名为 **Bob** 的 IAM 用户删除 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的签名证书。

```
Remove-IAMSigningCertificate -UserName Bob -CertificateId
 Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteSigningCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteUser** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteUser。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 创建只读和读写用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Delete an IAM user.
    /// </summary>
    /// <param name="userName">The username of the IAM user to delete.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserAsync(string userName)
    {
        var response = await _IAMService.DeleteUserAsync(new DeleteUserRequest
{ UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteUser。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_delete_user
#
# This function deletes the specified IAM user.
#
```

```
# Parameters:
#       -u user_name  -- The name of the user to create.
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_delete_user() {
  local user_name response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_delete_user"
    echo "Deletes an AWS Identity and Access Management (IAM) user. You must
 supply a username:"
    echo "  -u user_name    The name of the user."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  iecho "Parameters:\n"
```

```
  iecho "    User name:   $user_name"
  iecho ""

  # If the user does not exist, we don't want to try to delete it.
  if (! iam_user_exists "$user_name"); then
    errecho "ERROR: A user with that name does not exist in the account."
    return 1
  fi

  response=$(aws iam delete-user \
    --user-name "$user_name")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports delete-user operation failed.$response"
    return 1
  fi

  iecho "delete-user response:$response"
  iecho

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 DeleteUser。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DeleteUserRequest request;
```

```
    request.SetUserName(userName);
    auto outcome = iam.DeleteUser(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error deleting IAM user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;;
    }
    else {
        std::cout << "Successfully deleted IAM user " << userName << std::endl;
    }

    return outcome.IsSuccess();
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DeleteUser。

CLI

Amazon CLI

删除 IAM 用户

以下 `delete-user` 命令可将名为 Bob 的 IAM 用户从当前账户中删除。

```
aws iam delete-user \
    --user-name Bob
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的删除 IAM 用户。

- 有关 API 的详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteUser。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUser deletes a user.
func (wrapper UserWrapper) DeleteUser(ctx context.Context, userName string) error
 {
 _, err := wrapper.IamClient.DeleteUser(ctx, &iam.DeleteUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete user %v. Here's why: %v\n", userName, err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeleteUser。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.DeleteUserRequest;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <userName>\s

                Where:
                    userName - The name of the user to delete.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
```

```
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        deleteIAMUser(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void deleteIAMUser(IamClient iam, String userName) {
        try {
            DeleteUserRequest request = DeleteUserRequest.builder()
                    .userName(userName)
                    .build();

            iam.deleteUser(request);
            System.out.println("Successfully deleted IAM user " + userName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 DeleteUser。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

删除用户。

```
import { DeleteUserCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} name
 */
export const deleteUser = (name) => {
  const command = new DeleteUserCommand({ UserName: name });
  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-deleting-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteUser。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
};

iam.getUser(params, function (err, data) {
  if (err && err.code === "NoSuchEntity") {
    console.log("User " + process.argv[2] + " does not exist.");
  } else {
```

```
    iam.deleteUser(params, function (err, data) {
      if (err) {
        console.log("Error", err);
      } else {
        console.log("Success", data);
      }
    });
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-deleting-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DeleteUser。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun deleteIAMUser(userNameVal: String) {
    val request =
        DeleteUserRequest {
            userName = userNameVal
        }

    // To delete a user, ensure that the user's access keys are deleted first.
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.deleteUser(request)
        println("Successfully deleted user $userNameVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 DeleteUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除名为 **Bob** 的 IAM 用户。

```
Remove-IAMUser -UserName Bob
```

示例 2：此示例删除名为 **Theresa** 的 IAM 用户，以及必须先删除的所有元素。

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
```

```
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteUser。

Tools for PowerShell V5

示例 1：此示例删除名为 **Bob** 的 IAM 用户。

```
Remove-IAMUser -UserName Bob
```

示例 2：此示例删除名为 **Theresa** 的 IAM 用户，以及必须先删除的所有元素。

```
$name = "Theresa"

# find any groups and remove user from them
$groups = Get-IAMGroupForUser -UserName $name
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName $name -Force }

# find any inline policies and delete them
$inlinepols = Get-IAMUserPolicies -UserName $name
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 $name -Force}

# find any managed polices and detach them
$managedpols = Get-IAMAttachedUserPolicies -UserName $name
foreach ($pol in $managedpols) { Unregister-IAMUserPolicy -PolicyArn
 $pol.PolicyArn -UserName $name }

# find any signing certificates and delete them
$certs = Get-IAMSigningCertificate -UserName $name
```

```
foreach ($cert in $certs) { Remove-IAMSigningCertificate -CertificateId
 $cert.CertificateId -UserName $name -Force }

# find any access keys and delete them
$keys = Get-IAMAccessKey -UserName $name
foreach ($key in $keys) { Remove-IAMAccessKey -AccessKeyId $key.AccessKeyId -
UserName $name -Force }

# delete the user's login profile, if one exists - note: need to use try/catch to
 suppress not found error
try { $prof = Get-IAMLoginProfile -UserName $name -ea 0 } catch { out-null }
if ($prof) { Remove-IAMLoginProfile -UserName $name -Force }

# find any MFA device, detach it, and if virtual, delete it.
$mfa = Get-IAMMFADevice -UserName $name
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}

# finally, remove the user
Remove-IAMUser -UserName $name -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteUser。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.
```

```
    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 DeleteUser。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Error deleting user '#{user_name}': #{e.message}")
  end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteUser。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_user(client: &iamClient, user: &User) -> Result<(),
 SdkError<DeleteUserError>> {
    let user = user.clone();
    let mut tries: i32 = 0;
    let max_tries: i32 = 10;

    let response: Result<(), SdkError<DeleteUserError>> = loop {
        match client
            .delete_user()
            .user_name(user.user_name())
            .send()
            .await
        {
            Ok(_) => {
                break Ok(());
            }
            Err(e) => {
                tries += 1;
                if tries > max_tries {
                    break Err(e);
                }
                sleep(Duration::from_secs(2)).await;
            }
        }
```

```
    };

    response
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DeleteUser。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->deleteuser( iv_username = iv_user_name ).
        MESSAGE 'User deleted successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamdeleteconflictex.
        MESSAGE 'User cannot be deleted due to attached resources.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DeleteUser。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func deleteUser(user: IAMClientTypes.User) async throws {
        let input = DeleteUserInput(
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUser(input: input)
        } catch {
            print("ERROR: deleteUser:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DeleteUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteUserPermissionsBoundary** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteUserPermissionsBoundary。

CLI

Amazon CLI

删除 IAM 用户的权限边界

以下 delete-user-permissions-boundary 示例删除了附加到名为 intern 的 IAM 用户的权限边界。要对用户应用权限边界，请使用 put-user-permissions-boundary 命令。

```
aws iam delete-user-permissions-boundary \
    --user-name intern
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteUserPermissionsBoundary。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例展示了如何删除附加到 IAM 用户的权限边界。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteUserPermissionsBoundary。

Tools for PowerShell V5

示例 1：此示例展示了如何删除附加到 IAM 用户的权限边界。

```
Remove-IAMUserPermissionsBoundary -UserName joe
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteUserPermissionsBoundary。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteUserPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteUserPolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基本功能

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Delete an IAM user policy.
    /// </summary>
    /// <param name="policyName">The name of the IAM policy to delete.</param>
    /// <param name="userName">The username of the IAM user.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> DeleteUserPolicyAsync(string policyName, string
userName)
    {
        var response = await _IAMService.DeleteUserPolicyAsync(new
DeleteUserPolicyRequest { PolicyName = policyName, UserName = userName });

        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DeleteUserPolicy。

CLI

Amazon CLI

从 IAM 用户中移除策略

以下 `delete-user-policy` 命令可将指定策略从名为 Bob 的 IAM 用户中移除。

```
aws iam delete-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

此命令不生成任何输出。

要获取 IAM 用户的策略列表，请使用 `list-user-policies` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 账户中创建 IAM 用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteUserPolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// DeleteUserPolicy deletes an inline policy from a user.
func (wrapper UserWrapper) DeleteUserPolicy(ctx context.Context, userName string,
 policyName string) error {
```

```
 _, err := wrapper.IamClient.DeleteUserPolicy(ctx, &iam.DeleteUserPolicyInput{
  PolicyName: aws.String(policyName),
  UserName:   aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't delete policy from user %v. Here's why: %v\n", userName,
 err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DeleteUserPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除嵌入在名为 **Bob** 的 IAM 用户中的名为 **AccessToEC2Policy** 的内联策略。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

示例 2：此示例查找嵌入在名为 **Theresa** 的 IAM 用户中的所有内联策略，然后将其删除。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteUserPolicy。

Tools for PowerShell V5

示例 1：此示例删除嵌入在名为 **Bob** 的 IAM 用户中的名为 **AccessToEC2Policy** 的内联策略。

```
Remove-IAMUserPolicy -PolicyName AccessToEC2Policy -UserName Bob
```

示例 2：此示例查找嵌入在名为 **Theresa** 的 IAM 用户中的所有内联策略，然后将其删除。

```
$inlinepols = Get-IAMUserPolicies -UserName Theresa
foreach ($pol in $inlinepols) { Remove-IAMUserPolicy -PolicyName $pol -UserName
 Theresa -Force}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DeleteUserPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Deletes a user and their associated resources
 #
 # @param user_name [String] The name of the user to delete
 def delete_user(user_name)
   user = @iam_client.list_access_keys(user_name: user_name).access_key_metadata
   user.each do |key|
     @iam_client.delete_access_key({ access_key_id: key.access_key_id,
user_name: user_name })
     @logger.info("Deleted access key #{key.access_key_id} for user
'#{user_name}'.")
   end

   @iam_client.delete_user(user_name: user_name)
   @logger.info("Deleted user '#{user_name}'.")
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error deleting user '#{user_name}': #{e.message}")
 end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DeleteUserPolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn delete_user_policy(
    client: &iamClient,
    user: &User,
    policy_name: &str,
) -> Result<(), SdkError<DeleteUserPolicyError>> {
    client
        .delete_user_policy()
        .user_name(user.user_name())
        .policy_name(policy_name)
        .send()
        .await?;

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DeleteUserPolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
```

```
import AWSS3


    func deleteUserPolicy(user: IAMClientTypes.User, policyName: String) async
 throws {
        let input = DeleteUserPolicyInput(
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.deleteUserPolicy(input: input)
        } catch {
            print("ERROR: deleteUserPolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DeleteUserPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DeleteVirtualMfaDevice** 与 CLI 配合使用

以下代码示例演示如何使用 DeleteVirtualMfaDevice。

CLI

Amazon CLI

删除虚拟 MFA 设备

以下 delete-virtual-mfa-device 命令从当前账户中删除指定的 MFA 设备。

```
aws iam delete-virtual-mfa-device \
    --serial-number arn:aws:iam::123456789012:mfa/MFATest
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的停用 MFA 设备。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DeleteVirtualMfaDevice。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例删除 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 IAM 虚拟 MFA 设备。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

示例 2：此示例检查是否为 IAM 用户 Theresa 分配了 MFA 设备。如果找到设备，则会为 IAM 用户禁用该设备。如果设备是虚拟的，则会同时删除该设备。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DeleteVirtualMfaDevice。

Tools for PowerShell V5

示例 1：此示例删除 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 IAM 虚拟 MFA 设备。

```
Remove-IAMVirtualMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/bob
```

示例 2：此示例检查是否为 IAM 用户 Theresa 分配了 MFA 设备。如果找到设备，则会为 IAM 用户禁用该设备。如果设备是虚拟的，则会同时删除该设备。

```
$mfa = Get-IAMMFADevice -UserName Theresa
if ($mfa) {
    Disable-IAMMFADevice -SerialNumber $mfa.SerialNumber -UserName $name
    if ($mfa.SerialNumber -like "arn:*") { Remove-IAMVirtualMFADevice -
SerialNumber $mfa.SerialNumber }
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DeleteVirtualMfaDevice。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DetachGroupPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 DetachGroupPolicy。

CLI

Amazon CLI

从组中分离策略

此示例将从名为 Testers 的组重删除 ARN 为 arn:aws:iam::123456789012:policy/TesterAccessPolicy 的托管策略。

```
aws iam detach-group-policy \
    --group-name Testers \
    --policy-arn arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DetachGroupPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterAccessPolicy** 的托管组策略与名为 **Testers** 的组分离。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

示例 2：此示例查找附加到名为 **Testers** 的组的所有托管策略，并将其与该组分离。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DetachGroupPolicy。

Tools for PowerShell V5

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterAccessPolicy** 的托管组策略与名为 **Testers** 的组分离。

```
Unregister-IAMGroupPolicy -GroupName Testers -PolicyArn
 arn:aws:iam::123456789012:policy/TesterAccessPolicy
```

示例 2：此示例查找附加到名为 **Testers** 的组的所有托管策略，并将其与该组分离。

```
Get-IAMAttachedGroupPolicies -GroupName Testers | Unregister-IAMGroupPolicy -
Groupname Testers
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DetachGroupPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DetachRolePolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DetachRolePolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 了解基本功能
- 配置 Amazon ECS Service Connect
- 使用 Lambda 代理集成创建 REST API
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 Amazon Managed Grafana 工作区
- 创建第一个 Lambda 函数
- 开始使用 Amazon EKS

- Amazon MSK 入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- IoT Device Defender 入门
- Step Functions 入门
- 管理角色
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
   /// <summary>
   /// Detach an IAM policy from an IAM role.
   /// </summary>
   /// <param name="policyArn">The Amazon Resource Name (ARN) of the IAM
policy.</param>
   /// <param name="roleName">The name of the IAM role.</param>
   /// <returns>A Boolean value indicating the success of the action.</returns>
   public async Task<bool> DetachRolePolicyAsync(string policyArn, string
roleName)
   {
       var response = await _IAMService.DetachRolePolicyAsync(new
DetachRolePolicyRequest
       {
           PolicyArn = policyArn,
           RoleName = roleName,
       });
```

```
        return response.HttpStatusCode == System.Net.HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 DetachRolePolicy。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_detach_role_policy
#
# This function detaches an IAM policy to a tole.
#
# Parameters:
#       -n role_name -- The name of the IAM role.
#       -p policy_ARN -- The IAM policy document ARN..
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
##############################################################################
function iam_detach_role_policy() {
```

```
 local role_name policy_arn response
 local option OPTARG # Required to use getopts command in a function.

 # bashsupport disable=BP5008
 function usage() {
   echo "function iam_detach_role_policy"
   echo "Detaches an AWS Identity and Access Management (IAM) policy to an IAM
role."
   echo "  -n role_name   The name of the IAM role."
   echo "  -p policy_ARN -- The IAM policy document ARN."
   echo ""
 }

 # Retrieve the calling parameters.
 while getopts "n:p:h" option; do
   case "${option}" in
     n) role_name="${OPTARG}" ;;
     p) policy_arn="${OPTARG}" ;;
     h)
       usage
       return 0
       ;;
     \?)
       echo "Invalid parameter"
       usage
       return 1
       ;;
   esac
 done
 export OPTIND=1

 if [[ -z "$role_name" ]]; then
   errecho "ERROR: You must provide a role name with the -n parameter."
   usage
   return 1
 fi

 if [[ -z "$policy_arn" ]]; then
   errecho "ERROR: You must provide a policy ARN with the -p parameter."
   usage
   return 1
 fi

 response=$(aws iam detach-role-policy \
```

```
    --role-name "$role_name" \
    --policy-arn "$policy_arn")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports detach-role-policy operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 DetachRolePolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::DetachRolePolicyRequest detachRequest;
    detachRequest.SetRoleName(roleName);
    detachRequest.SetPolicyArn(policyArn);

    auto detachOutcome = iam.DetachRolePolicy(detachRequest);
    if (!detachOutcome.IsSuccess()) {
        std::cerr << "Failed to detach policy " << policyArn << " from role "
                  << roleName << ": " << detachOutcome.GetError().GetMessage() <<
                  std::endl;
    }
    else {
```

```
        std::cout << "Successfully detached policy " << policyArn << " from role
"
                  << roleName << std::endl;
    }

    return detachOutcome.IsSuccess();
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 DetachRolePolicy。

CLI

Amazon CLI

从角色分离策略

此示例将从名为 FedTesterRole 的角色删除具有 ARN arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy 的托管策略。

```
aws iam detach-role-policy \
    --role-name FedTesterRole \
    --policy-arn arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DetachRolePolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// DetachRolePolicy detaches a policy from a role.
func (wrapper RoleWrapper) DetachRolePolicy(ctx context.Context, roleName string,
 policyArn string) error {
 _, err := wrapper.IamClient.DetachRolePolicy(ctx, &iam.DetachRolePolicyInput{
  PolicyArn: aws.String(policyArn),
  RoleName:  aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't detach policy from role %v. Here's why: %v\n", roleName,
 err)
 }
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 DetachRolePolicy。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.DetachRolePolicyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DetachRolePolicy {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleName> <policyArn>\s

                Where:
                    roleName - A role name that you can obtain from the AWS
 Management Console.\s
                    policyArn - A policy ARN that you can obtain from the AWS
 Management Console.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String roleName = args[0];
        String policyArn = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();
        detachPolicy(iam, roleName, policyArn);
        System.out.println("Done");
        iam.close();
    }

    public static void detachPolicy(IamClient iam, String roleName, String
 policyArn) {
        try {
            DetachRolePolicyRequest request = DetachRolePolicyRequest.builder()
                    .roleName(roleName)
                    .policyArn(policyArn)
                    .build();

            iam.detachRolePolicy(request);
            System.out.println("Successfully detached policy " + policyArn +
                    " from role " + roleName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 DetachRolePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

分离策略。

```
import { DetachRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 * @param {string} roleName
 */
export const detachRolePolicy = (policyArn, roleName) => {
  const command = new DetachRolePolicyCommand({
    PolicyArn: policyArn,
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-policies.html#iam-examples-policies-detaching-role-policy。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DetachRolePolicy。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var paramsRoleList = {
  RoleName: process.argv[2],
};

iam.listAttachedRolePolicies(paramsRoleList, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var myRolePolicies = data.AttachedPolicies;
    myRolePolicies.forEach(function (val, index, array) {
      if (myRolePolicies[index].PolicyName === "AmazonDynamoDBFullAccess") {
        var params = {
          PolicyArn: "arn:aws:iam::aws:policy/AmazonDynamoDBFullAccess",
          RoleName: process.argv[2],
        };
        iam.detachRolePolicy(params, function (err, data) {
          if (err) {
            console.log("Unable to detach policy from role", err);
          } else {
            console.log("Policy detached from role successfully");
            process.exit();
          }
        });
      }
    });
  }
```

```
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-policies.html#iam-examples-policies-detaching-role-policy。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 DetachRolePolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun detachPolicy(
    roleNameVal: String,
    policyArnVal: String,
) {
    val request =
        DetachRolePolicyRequest {
            roleName = roleNameVal
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.detachRolePolicy(request)
        println("Successfully detached policy $policyArnVal from role
 $roleNameVal")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 DetachRolePolicy

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** 的托管组策略与名为 **FedTesterRole** 的角色分离。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

示例 2：此示例查找附加到名为 **FedTesterRole** 的角色的所有托管策略，并将其与该角色分离。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DetachRolePolicy。

Tools for PowerShell V5

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy** 的托管组策略与名为 **FedTesterRole** 的角色分离。

```
Unregister-IAMRolePolicy -RoleName FedTesterRole -PolicyArn
 arn:aws:iam::123456789012:policy/FederatedTesterAccessPolicy
```

示例 2：此示例查找附加到名为 **FedTesterRole** 的角色的所有托管策略，并将其与该角色分离。

```
Get-IAMAttachedRolePolicyList -RoleName FedTesterRole | Unregister-IAMRolePolicy
 -Rolename FedTesterRole
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DetachRolePolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

使用 Boto3 策略对象从角色分离策略。

```
def detach_from_role(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Policy(policy_arn).detach_role(RoleName=role_name)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

使用 Boto3 角色对象从角色分离策略。

```
def detach_policy(role_name, policy_arn):
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
```

```
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 DetachRolePolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
```

```
 # @return [String] The policy ARN if successful, otherwise nil
 def create_policy(policy_name, policy_document)
   response = @iam_client.create_policy(
     policy_name: policy_name,
     policy_document: policy_document.to_json
   )
   response.policy.arn
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error creating policy: #{e.message}")
   nil
 end

 # Fetches an IAM policy by its ARN
 # @param policy_arn [String] the ARN of the IAM policy to retrieve
 # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
 def get_policy(policy_arn)
   response = @iam_client.get_policy(policy_arn: policy_arn)
   policy = response.policy
   @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
   policy
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
   raise
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
   raise
 end

 # Attaches a policy to a role
 #
 # @param role_name [String] The name of the role
 # @param policy_arn [String] The policy ARN
 # @return [Boolean] true if successful, false otherwise
 def attach_policy_to_role(role_name, policy_arn)
   @iam_client.attach_role_policy(
     role_name: role_name,
     policy_arn: policy_arn
   )
   true
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error attaching policy to role: #{e.message}")
```

```
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DetachRolePolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn detach_role_policy(
    client: &iamClient,
    role_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_role_policy()
        .role_name(role_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DetachRolePolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
```

```
        lo_iam->detachrolepolicy(
          iv_rolename = iv_role_name
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Policy detached from role successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role or policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DetachRolePolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func detachRolePolicy(policy: IAMClientTypes.Policy, role:
 IAMClientTypes.Role) async throws {
        let input = DetachRolePolicyInput(
            policyArn: policy.arn,
            roleName: role.roleName
        )

        do {
            _ = try await iamClient.detachRolePolicy(input: input)
        } catch {
            print("ERROR: detachRolePolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 DetachRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DetachUserPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DetachUserPolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 创建只读和读写用户

CLI

Amazon CLI

从用户分离策略

此示例将从用户 Bob 删除具有 ARN arn:aws:iam::123456789012:policy/TesterPolicy 的托管策略。

```
aws iam detach-user-policy \
    --user-name Bob \
    --policy-arn arn:aws:iam::123456789012:policy/TesterPolicy
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的更改 IAM 用户的权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DetachUserPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterPolicy** 的托管策略与名为 **Bob** 的 IAM 用户分离。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

示例 2：此示例查找附加到名为 **Theresa** 的 IAM 用户的所有托管策略，并将这些策略与该用户分离。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DetachUserPolicy。

Tools for PowerShell V5

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/TesterPolicy** 的托管策略与名为 **Bob** 的 IAM 用户分离。

```
Unregister-IAMUserPolicy -UserName Bob -PolicyArn
 arn:aws:iam::123456789012:policy/TesterPolicy
```

示例 2：此示例查找附加到名为 **Theresa** 的 IAM 用户的所有托管策略，并将这些策略与该用户分离。

```
Get-IAMAttachedUserPolicyList -UserName Theresa | Unregister-IAMUserPolicy -
Username Theresa
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 DetachUserPolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 DetachUserPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Detaches a policy from a user
 #
 # @param user_name [String] The name of the user
 # @param policy_arn [String] The ARN of the policy to detach
 # @return [Boolean] true if the policy was successfully detached, false
otherwise
 def detach_user_policy(user_name, policy_arn)
   @iam_client.detach_user_policy(
     user_name: user_name,
```

```
      policy_arn: policy_arn
    )
    @logger.info("Policy '#{policy_arn}' detached from user '#{user_name}'
successfully.")
    true
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error('Error detaching policy: Policy or user does not exist.')
    false
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from user '#{user_name}':
#{e.message}")
    false
  end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 DetachUserPolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn detach_user_policy(
    client: &iamClient,
    user_name: &str,
    policy_arn: &str,
) -> Result<(), iamError> {
    client
        .detach_user_policy()
        .user_name(user_name)
        .policy_arn(policy_arn)
        .send()
        .await?;

    Ok(())
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 DetachUserPolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    lo_iam->detachuserpolicy(
      iv_username = iv_user_name
      iv_policyarn = iv_policy_arn ).
    MESSAGE 'Policy detached from user successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'User or policy does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 DetachUserPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **EnableMfaDevice** 与 CLI 配合使用

以下代码示例演示如何使用 EnableMfaDevice。

CLI

Amazon CLI

启用 MFA 设备

使用 `create-virtual-mfa-device` 命令创建新的虚拟 MFA 设备后，您可以将 MFA 设备分配给用户。以下 `enable-mfa-device` 示例将序列号为 `arn:aws:iam::210987654321:mfa/BobsMFADevice` 的 MFA 设备分配给用户 Bob。该命令还通过按顺序包含虚拟 MFA 设备中的前两个代码，将设备与 Amazon 同步。

```
aws iam enable-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 789012
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的启用虚拟多重身份验证（MFA）设备。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 EnableMfaDevice。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令启用序列号为 **987654321098** 的硬件 MFA 设备，并将该设备与用户 **Bob** 关联。它包含设备中按顺序排列的前两个代码。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

示例 2：此示例创建并启用虚拟 MFA 设备。第一条命令创建虚拟设备并在变量 **$MFADevice** 中返回设备的对象表示形式。您可以使用 **.Base32StringSeed** 或 **QRCodePng** 属性来配置用户的软件应用程序。最后一条命令将该设备分配给用户 **David**，通过其序列号识别设备。该命令还通过按顺序包含虚拟 MFA 设备中的前两个代码，将设备与 Amazon 同步。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
```

```
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 EnableMfaDevice。

Tools for PowerShell V5

示例 1：此命令启用序列号为 **987654321098** 的硬件 MFA 设备，并将该设备与用户 **Bob** 关联。它包含设备中按顺序排列的前两个代码。

```
Enable-IAMMFADevice -UserName "Bob" -SerialNumber "987654321098" -
AuthenticationCode1 "12345678" -AuthenticationCode2 "87654321"
```

示例 2：此示例创建并启用虚拟 MFA 设备。第一条命令创建虚拟设备并在变量 **$MFADevice** 中返回设备的对象表示形式。您可以使用 **.Base32StringSeed** 或 **QRCodePng** 属性来配置用户的软件应用程序。最后一条命令将该设备分配给用户 **David**，通过其序列号识别设备。该命令还通过按顺序包含虚拟 MFA 设备中的前两个代码，将设备与 Amazon 同步。

```
$MFADevice = New-IAMVirtualMFADevice -VirtualMFADeviceName "MyMFADevice"
# see example for New-IAMVirtualMFADevice to see how to configure the software
 program with PNG or base32 seed code
Enable-IAMMFADevice -UserName "David" -SerialNumber -SerialNumber
 $MFADevice.SerialNumber -AuthenticationCode1 "24681357" -AuthenticationCode2
 "13572468"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 EnableMfaDevice。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GenerateCredentialReport** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GenerateCredentialReport。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

CLI

Amazon CLI

生成凭证报告

以下示例尝试为 Amazon 账户生成凭证报告。

```
aws iam generate-credential-report
```

输出：

```
{
    "State":  "STARTED",
    "Description": "No report exists. Starting a new report generation task"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的获取 Amazon 账户的凭证报告。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GenerateCredentialReport。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例请求生成一份新报告，报告可每四小时生成一次。如果上次报告仍然是最新的，则“状态”字段显示为 **COMPLETE**。使用 **Get-IAMCredentialReport** 查看已完成的报告。

```
Request-IAMCredentialReport
```

输出：

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GenerateCredentialReport。

Tools for PowerShell V5

示例 1：此示例请求生成一份新报告，报告可每四小时生成一次。如果上次报告仍然是最新的，则“状态”字段显示为 **COMPLETE**。使用 **Get-IAMCredentialReport** 查看已完成的报告。

```
Request-IAMCredentialReport
```

输出：

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GenerateCredentialReport。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def generate_credential_report():
    """
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
```

```
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
account.")
        raise
    else:
        return response
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GenerateCredentialReport。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->generatecredentialreport( ).
        MESSAGE 'Credential report generation started.' TYPE 'I'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Report generation limit exceeded.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when generating credential report.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GenerateCredentialReport。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GenerateServiceLastAccessedDetails** 与 CLI 配合使用

以下代码示例演示如何使用 GenerateServiceLastAccessedDetails。

CLI

Amazon CLI

示例 1：为自定义策略生成服务访问报告

以下 generate-service-last-accessed-details 示例启动后台作业以生成一份报告，其中列出使用名为 intern-boundary 的自定义策略的 IAM 用户和其他实体所访问的服务。创建报告后，您可以通过运行 get-service-last-accessed-details 命令来显示该报告。

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::123456789012:policy/intern-boundary
```

输出：

```
{
    "JobId": "2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc"
}
```

示例 2：为 Amazon 托管 AdministratorAccess 策略生成服务访问报告

以下 generate-service-last-accessed-details 示例启动后台作业以生成一份报告，其中列出使用 Amazon 托管 AdministratorAccess 策略的 IAM 用户和其他实体所访问的服务。创建报告后，您可以通过运行 get-service-last-accessed-details 命令来显示该报告。

```
aws iam generate-service-last-accessed-details \
    --arn arn:aws:iam::aws:policy/AdministratorAccess
```

输出：

```
{
    "JobId": "78b6c2ba-d09e-6xmp-7039-ecde30b26916"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用上次访问的信息优化 Amazon 中的权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GenerateServiceLastAccessedDetails。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例等同于 GenerateServiceLastAccessedDetails API 的 cmdlet。这提供了一个可在 Get-IAMServiceLastAccessedDetail 和 Get-IAMServiceLastAccessedDetailWithEntity 中使用的作业 ID

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GenerateServiceLastAccessedDetails。

Tools for PowerShell V5

示例 1：此示例等同于 GenerateServiceLastAccessedDetails API 的 cmdlet。这提供了一个可在 Get-IAMServiceLastAccessedDetail 和 Get-IAMServiceLastAccessedDetailWithEntity 中使用的作业 ID

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GenerateServiceLastAccessedDetails。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetAccessKeyLastUsed** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetAccessKeyLastUsed。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问密钥

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::accessKeyLastUsed(const Aws::String &secretKeyID,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetAccessKeyLastUsedRequest request;

    request.SetAccessKeyId(secretKeyID);

    Aws::IAM::Model::GetAccessKeyLastUsedOutcome outcome =
 iam.GetAccessKeyLastUsed(
            request);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error querying last used time for access key " <<
                  secretKeyID << ":" << outcome.GetError().GetMessage() <<
 std::endl;
    }
    else {
        Aws::String lastUsedTimeString =
                outcome.GetResult()
                        .GetAccessKeyLastUsed()
                        .GetLastUsedDate()
                        .ToGmtString(Aws::Utils::DateFormat::ISO_8601);
        std::cout << "Access key " << secretKeyID << " last used at time " <<
                  lastUsedTimeString << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 GetAccessKeyLastUsed。

CLI

Amazon CLI

检索上次使用指定访问密钥的时间信息

以下示例将检索上次使用访问密钥 ABCDEXAMPLE 的时间信息。

```
aws iam get-access-key-last-used \
    --access-key-id ABCDEXAMPLE
```

输出：

```
{
    "UserName":  "Bob",
    "AccessKeyLastUsed": {
        "Region": "us-east-1",
        "ServiceName": "iam",
        "LastUsedDate": "2015-06-16T22:45:00Z"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的访问密钥。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetAccessKeyLastUsed。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

获取访问密钥。

```
import { GetAccessKeyLastUsedCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} accessKeyId
 */
export const getAccessKeyLastUsed = async (accessKeyId) => {
  const command = new GetAccessKeyLastUsedCommand({
    AccessKeyId: accessKeyId,
  });

  const response = await client.send(command);

  if (response.AccessKeyLastUsed?.LastUsedDate) {
    console.log(`
    ${accessKeyId} was last used by ${response.UserName} via
    the ${response.AccessKeyLastUsed.ServiceName} service on
    ${response.AccessKeyLastUsed.LastUsedDate.toISOString()}
    `);
  }

  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-last-used。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetAccessKeyLastUsed。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getAccessKeyLastUsed(
  { AccessKeyId: "ACCESS_KEY_ID" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data.AccessKeyLastUsed);
    }
  }
);
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-last-used。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetAccessKeyLastUsed。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回所提供访问密钥的拥有用户名和上次使用信息。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetAccessKeyLastUsed。

Tools for PowerShell V5

示例 1：返回所提供访问密钥的拥有用户名和上次使用信息。

```
Get-IAMAccessKeyLastUsed -AccessKeyId ABCDEXAMPLE
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetAccessKeyLastUsed。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetAccessKeyLastUsed。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->getaccesskeylastused(
      iv_accesskeyid = iv_access_key_id ).
    MESSAGE 'Retrieved access key last used information.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetAccessKeyLastUsed。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetAccountAuthorizationDetails** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetAccountAuthorizationDetails。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

CLI

Amazon CLI

列出 Amazon 账户 IAM 用户、组、角色和策略

以下 `get-account-authorization-details` 命令将返回 Amazon 账户中所有 IAM 用户、组、角色和策略的信息。

```
aws iam get-account-authorization-details
```

输出：

```
{
    "RoleDetailList": [
        {
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "RoleId": "AROA1234567890EXAMPLE",
            "CreateDate": "2014-07-30T17:09:20Z",
            "InstanceProfileList": [
                {
                    "InstanceProfileId": "AIPA1234567890EXAMPLE",
                    "Roles": [
                        {
                            "AssumeRolePolicyDocument": {
                                "Version":"2012-10-17",
                                "Statement": [
                                    {
                                        "Sid": "",
                                        "Effect": "Allow",
                                        "Principal": {
```

```
                                        "Service": "ec2.amazonaws.com"
                                    },
                                    "Action": "sts:AssumeRole"
                                }
                            ]
                        },
                        "RoleId": "AROA1234567890EXAMPLE",
                        "CreateDate": "2014-07-30T17:09:20Z",
                        "RoleName": "EC2role",
                        "Path": "/",
                        "Arn": "arn:aws:iam::123456789012:role/EC2role"
                    }
                ],
                "CreateDate": "2014-07-30T17:09:20Z",
                "InstanceProfileName": "EC2role",
                "Path": "/",
                "Arn": "arn:aws:iam::123456789012:instance-profile/EC2role"
            }
        ],
        "RoleName": "EC2role",
        "Path": "/",
        "AttachedManagedPolicies": [
            {
                "PolicyName": "AmazonS3FullAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AmazonS3FullAccess"
            },
            {
                "PolicyName": "AmazonDynamoDBFullAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/
AmazonDynamoDBFullAccess"
            }
        ],
        "RoleLastUsed": {
            "Region": "us-west-2",
            "LastUsedDate": "2019-11-13T17:30:00Z"
        },
        "RolePolicyList": [],
        "Arn": "arn:aws:iam::123456789012:role/EC2role"
    }
],
"GroupDetailList": [
    {
        "GroupId": "AIDA1234567890EXAMPLE",
        "AttachedManagedPolicies": {
```

```
                "PolicyName": "AdministratorAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
            },
            "GroupName": "Admins",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "CreateDate": "2013-10-14T18:32:24Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": {
                "PolicyName": "PowerUserAccess",
                "PolicyArn": "arn:aws:iam::aws:policy/PowerUserAccess"
            },
            "GroupName": "Dev",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Dev",
            "CreateDate": "2013-10-14T18:33:55Z",
            "GroupPolicyList": []
        },
        {
            "GroupId": "AIDA1234567890EXAMPLE",
            "AttachedManagedPolicies": [],
            "GroupName": "Finance",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:group/Finance",
            "CreateDate": "2013-10-14T18:57:48Z",
            "GroupPolicyList": [
                {
                    "PolicyName": "policygen-201310141157",
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Action": "aws-portal:*",
                                "Sid": "Stmt1381777017000",
                                "Resource": "*",
                                "Effect": "Allow"
                            }
                        ]
                    }
                }
            ]
```

```
        }
    ],
    "UserDetailList": [
        {
            "UserName": "Alice",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:24Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Alice"
        },
        {
            "UserName": "Bob",
            "GroupList": [
                "Admins"
            ],
            "CreateDate": "2013-10-14T18:32:25Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [
                {
                    "PolicyName": "DenyBillingAndIAMPolicy",
                    "PolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Deny",
                            "Action": [
                                "aws-portal:*",
                                "iam:*"
                            ],
                            "Resource": "*"
                        }
                    }
                }
            ],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        },
        {
            "UserName": "Charlie",
```

```
            "GroupList": [
                "Dev"
            ],
            "CreateDate": "2013-10-14T18:33:56Z",
            "UserId": "AIDA1234567890EXAMPLE",
            "UserPolicyList": [],
            "Path": "/",
            "AttachedManagedPolicies": [],
            "Arn": "arn:aws:iam::123456789012:user/Charlie"
        }
    ],
    "Policies": [
        {
            "PolicyName": "create-update-delete-set-managed-policies",
            "CreateDate": "2015-02-06T19:58:34Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-02-06T19:58:34Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": {
                            "Effect": "Allow",
                            "Action": [
                                "iam:CreatePolicy",
                                "iam:CreatePolicyVersion",
                                "iam:DeletePolicy",
                                "iam:DeletePolicyVersion",
                                "iam:GetPolicy",
                                "iam:GetPolicyVersion",
                                "iam:ListPolicies",
                                "iam:ListPolicyVersions",
                                "iam:SetDefaultPolicyVersion"
                            ],
                            "Resource": "*"
                        }
                    },
                    "IsDefaultVersion": true
                }
            ],
```

```
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/create-update-delete-set-
managed-policies",
            "UpdateDate": "2015-02-06T19:58:34Z"
        },
        {
            "PolicyName": "S3-read-only-specific-bucket",
            "CreateDate": "2015-01-21T21:39:41Z",
            "AttachmentCount": 1,
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2015-01-21T21:39:41Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Action": [
                                    "s3:Get*",
                                    "s3:List*"
                                ],
                                "Resource": [
                                    "arn:aws:s3:::amzn-s3-demo-bucket",
                                    "arn:aws:s3:::amzn-s3-demo-bucket/*"
                                ]
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:policy/S3-read-only-specific-
bucket",
            "UpdateDate": "2015-01-21T23:39:41Z"
        },
        {
            "PolicyName": "AmazonEC2FullAccess",
            "CreateDate": "2015-02-06T18:40:15Z",
            "AttachmentCount": 1,
```

```
            "IsAttachable": true,
            "PolicyId": "ANPA1234567890EXAMPLE",
            "DefaultVersionId": "v1",
            "PolicyVersionList": [
                {
                    "CreateDate": "2014-10-30T20:59:46Z",
                    "VersionId": "v1",
                    "Document": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Action": "ec2:*",
                                "Effect": "Allow",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "elasticloadbalancing:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "cloudwatch:*",
                                "Resource": "*"
                            },
                            {
                                "Effect": "Allow",
                                "Action": "autoscaling:*",
                                "Resource": "*"
                            }
                        ]
                    },
                    "IsDefaultVersion": true
                }
            ],
            "Path": "/",
            "Arn": "arn:aws:iam::aws:policy/AmazonEC2FullAccess",
            "UpdateDate": "2015-02-06T18:40:15Z"
        }
    ],
    "Marker": "EXAMPLEkakv9BCuUNFDtxWSyfzetYwEx2ADc8dnzfvERF5S6YMvXKx41t6gCl/
eeaCX3Jo94/bKqezEAg8TEVS99EKFLxm3jtbpl25FDWEXAMPLE",
    "IsTruncated": true
```

```
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 Amazon 安全审计指南。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetAccountAuthorizationDetails。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例获取有关 Amazon 账户中身份的授权详细信息，并显示返回对象的元素列表，括用户、组和角色。例如，**UserDetailList** 属性显示有关用户的详细信息。**RoleDetailList** 和 **GroupDetailList** 属性中也提供类似的信息。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

输出：

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

输出：

```
Arn             : arn:aws:iam::123456789012:user/Administrator
CreateDate      : 10/16/2014 9:03:09 AM
GroupList       : {Administrators}
Path            : /
UserId          : AIDACKCEVSQ6CEXAMPLE1
UserName        : Administrator
UserPolicyList : {}

Arn             : arn:aws:iam::123456789012:user/Bob
CreateDate      : 4/6/2015 12:54:42 PM
```

```
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/BackupToS3
CreateDate     : 1/27/2015 10:15:08 AM
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetAccountAuthorizationDetails。

Tools for PowerShell V5

示例 1：此示例获取有关 Amazon 账户中身份的授权详细信息，并显示返回对象的元素列表，括用户、组和角色。例如，**UserDetailList** 属性显示有关用户的详细信息。**RoleDetailList** 和 **GroupDetailList** 属性中也提供类似的信息。

```
$Details=Get-IAMAccountAuthorizationDetail
$Details
```

输出：

```
GroupDetailList : {Administrators, Developers, Testers, Backup}
IsTruncated     : False
Marker          :
RoleDetailList  : {TestRole1, AdminRole, TesterRole, clirole...}
UserDetailList  : {Administrator, Bob, BackupToS3, }
```

```
$Details.UserDetailList
```

输出：

```
Arn            : arn:aws:iam::123456789012:user/Administrator
CreateDate     : 10/16/2014 9:03:09 AM
GroupList      : {Administrators}
```

```
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE1
UserName       : Administrator
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/Bob
CreateDate     : 4/6/2015 12:54:42 PM
GroupList      : {Developers}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE2
UserName       : bab
UserPolicyList : {}

Arn            : arn:aws:iam::123456789012:user/BackupToS3
CreateDate     : 1/27/2015 10:15:08 AM
GroupList      : {Backup}
Path           : /
UserId         : AIDACKCEVSQ6CEXAMPLE3
UserName       : BackupToS3
UserPolicyList : {BackupServicePermissionsToS3Buckets}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetAccountAuthorizationDetails。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
```

```
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetAccountAuthorizationDetails。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   TRY.
       oo_result = lo_iam->getaccountauthdetails( ).
       MESSAGE 'Retrieved account authorization details.' TYPE 'I'.
     CATCH /aws1/cx_iamservicefailureex.
       MESSAGE 'Service failure when getting account authorization details.'
TYPE 'E'.
   ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetAccountAuthorizationDetails。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetAccountPasswordPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetAccountPasswordPolicy。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// Gets the IAM password policy for an AWS account.
    /// </summary>
    /// <returns>The PasswordPolicy for the AWS account.</returns>
    public async Task<PasswordPolicy> GetAccountPasswordPolicyAsync()
    {
        var response = await _IAMService.GetAccountPasswordPolicyAsync(new
GetAccountPasswordPolicyRequest());
        return response.PasswordPolicy;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 GetAccountPasswordPolicy。

CLI

Amazon CLI

查看当前账户密码策略

以下 get-account-password-policy 命令将显示有关当前账户密码策略的详细信息。

```
aws iam get-account-password-policy
```

输出：

```
{
    "PasswordPolicy": {
        "AllowUsersToChangePassword": false,
        "RequireLowercaseCharacters": false,
        "RequireUppercaseCharacters": false,
        "MinimumPasswordLength": 8,
        "RequireNumbers": true,
        "RequireSymbols": true
    }
}
```

如果没有为账户定义密码策略，命令将返回 NoSuchEntity 错误。

有关更多信息，请参阅《Amazon IAM 用户指南》中的为 IAM 用户设置账户密码策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetAccountPasswordPolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)
```

```
// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// GetAccountPasswordPolicy gets the account password policy for the current
 account.
// If no policy has been set, a NoSuchEntityException is error is returned.
func (wrapper AccountWrapper) GetAccountPasswordPolicy(ctx context.Context)
 (*types.PasswordPolicy, error) {
 var pwPolicy *types.PasswordPolicy
 result, err := wrapper.IamClient.GetAccountPasswordPolicy(ctx,
  &iam.GetAccountPasswordPolicyInput{})
 if err != nil {
  log.Printf("Couldn't get account password policy. Here's why: %v\n", err)
 } else {
  pwPolicy = result.PasswordPolicy
 }
 return pwPolicy, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 GetAccountPasswordPolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

获取账户密码策略。

```
import {
  GetAccountPasswordPolicyCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const getAccountPasswordPolicy = async () => {
  const command = new GetAccountPasswordPolicyCommand({});

  const response = await client.send(command);
  console.log(response.PasswordPolicy);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetAccountPasswordPolicy。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getAccountPasswordPolicy()
    {
        return $this->iamClient->getAccountPasswordPolicy();
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 GetAccountPasswordPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回有关当前账户密码策略的详细信息。如果没有为账户定义密码策略，则命令将返回 **NoSuchEntity** 错误。

```
Get-IAMAccountPasswordPolicy
```

输出：

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetAccountPasswordPolicy。

Tools for PowerShell V5

示例 1：此示例返回有关当前账户密码策略的详细信息。如果没有为账户定义密码策略，则命令将返回 **NoSuchEntity** 错误。

```
Get-IAMAccountPasswordPolicy
```

输出：

```
AllowUsersToChangePassword : True
ExpirePasswords            : True
HardExpiry                 : False
MaxPasswordAge             : 90
MinimumPasswordLength      : 8
PasswordReusePrevention    : 20
RequireLowercaseCharacters : True
```

```
RequireNumbers             : True
RequireSymbols             : False
RequireUppercaseCharacters : True
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetAccountPasswordPolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def print_password_policy():
    """
    Prints the password policy for the account.
    """
    try:
        pw_policy = iam.AccountPasswordPolicy()
        print("Current account password policy:")
        print(
            f"\tallow_users_to_change_password:
 {pw_policy.allow_users_to_change_password}"
        )
        print(f"\texpire_passwords: {pw_policy.expire_passwords}")
        print(f"\thard_expiry: {pw_policy.hard_expiry}")
        print(f"\tmax_password_age: {pw_policy.max_password_age}")
        print(f"\tminimum_password_length: {pw_policy.minimum_password_length}")
        print(f"\tpassword_reuse_prevention:
 {pw_policy.password_reuse_prevention}")
        print(
            f"\trequire_lowercase_characters:
 {pw_policy.require_lowercase_characters}"
        )
        print(f"\trequire_numbers: {pw_policy.require_numbers}")
        print(f"\trequire_symbols: {pw_policy.require_symbols}")
        print(
```

```
            f"\trequire_uppercase_characters:
 {pw_policy.require_uppercase_characters}"
        )
        printed = True
    except ClientError as error:
        if error.response["Error"]["Code"] == "NoSuchEntity":
            print("The account does not have a password policy set.")
        else:
            logger.exception("Couldn't get account password policy.")
            raise
    else:
        return printed
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetAccountPasswordPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Class to manage IAM account password policies
class PasswordPolicyManager
  attr_accessor :iam_client, :logger

  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'IAMPolicyManager'
  end

  # Retrieves and logs the account password policy
  def print_account_password_policy
```

```
    response = @iam_client.get_account_password_policy
    @logger.info("The account password policy is:
 #{response.password_policy.to_h}")
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.info('The account does not have a password policy.')
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't print the account password policy. Error: #{e.code} -
 #{e.message}")
    raise
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 GetAccountPasswordPolicy。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn get_account_password_policy(
    client: &iamClient,
) -> Result<GetAccountPasswordPolicyOutput,
 SdkError<GetAccountPasswordPolicyError>> {
    let response = client.get_account_password_policy().send().await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 GetAccountPasswordPolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->getaccountpasswordpolicy( ).
    MESSAGE 'Retrieved account password policy.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'No password policy exists.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting password policy.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetAccountPasswordPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`GetAccountSummary`** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `GetAccountSummary`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

CLI

Amazon CLI

获取当前账户中 IAM 实体使用情况和 IAM 配额的信息

以下 get-account-summary 命令将返回账户中当前 IAM 实体使用情况和当前 IAM 实体配额的信息。

```
aws iam get-account-summary
```

输出：

```
{
    "SummaryMap": {
        "UsersQuota": 5000,
        "GroupsQuota": 100,
        "InstanceProfiles": 6,
        "SigningCertificatesPerUserQuota": 2,
        "AccountAccessKeysPresent": 0,
        "RolesQuota": 250,
        "RolePolicySizeQuota": 10240,
        "AccountSigningCertificatesPresent": 0,
        "Users": 27,
        "ServerCertificatesQuota": 20,
        "ServerCertificates": 0,
        "AssumeRolePolicySizeQuota": 2048,
        "Groups": 7,
        "MFADevicesInUse": 1,
        "Roles": 3,
        "AccountMFAEnabled": 1,
        "MFADevices": 3,
        "GroupsPerUserQuota": 10,
        "GroupPolicySizeQuota": 5120,
        "InstanceProfilesQuota": 100,
        "AccessKeysPerUserQuota": 2,
        "Providers": 0,
        "UserPolicySizeQuota": 2048
    }
}
```

有关实体限制的更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 和 Amazon STS 配额。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetAccountSummary。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将返回有关 Amazon Web Services 账户 中当前 IAM 实体使用情况和当前 IAM 实体配额的信息。

```
Get-IAMAccountSummary
```

输出：

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
InstanceProfilesQuota                   100
MFADevices                              4
AccessKeysPerUserQuota                  2
AttachedPoliciesPerUserQuota            2
SigningCertificatesPerUserQuota         2
```

```
PolicyVersionsInUse                     4
InstanceProfiles                        1
...
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetAccountSummary。

Tools for PowerShell V5

示例 1：此示例将返回有关 Amazon Web Services 账户 中当前 IAM 实体使用情况和当前 IAM 实体配额的信息。

```
Get-IAMAccountSummary
```

输出：

```
Key                                     Value
Users                                   7
GroupPolicySizeQuota                    5120
PolicyVersionsInUseQuota                10000
ServerCertificatesQuota                 20
AccountSigningCertificatesPresent       0
AccountAccessKeysPresent                0
Groups                                  3
UsersQuota                              5000
RolePolicySizeQuota                     10240
UserPolicySizeQuota                     2048
GroupsPerUserQuota                      10
AssumeRolePolicySizeQuota               2048
AttachedPoliciesPerGroupQuota           2
Roles                                   9
VersionsPerPolicyQuota                  5
GroupsQuota                             100
PolicySizeQuota                         5120
Policies                                5
RolesQuota                              250
ServerCertificates                      0
AttachedPoliciesPerRoleQuota            2
MFADevicesInUse                         2
PoliciesQuota                           1000
AccountMFAEnabled                       1
Providers                               2
```

```
InstanceProfilesQuota                    100
MFADevices                               4
AccessKeysPerUserQuota                   2
AttachedPoliciesPerUserQuota             2
SigningCertificatesPerUserQuota          2
PolicyVersionsInUse                      4
InstanceProfiles                         1
...
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetAccountSummary。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_summary():
    """
    Gets a summary of account usage.

    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetAccountSummary。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->getaccountsummary( ).
    MESSAGE 'Retrieved account summary.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when getting account summary.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetAccountSummary。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetContextKeysForCustomPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 GetContextKeysForCustomPolicy。

CLI

Amazon CLI

示例 1：列出作为命令行参数提供的一个或多个自定义 JSON 策略所引用的上下文键

以下 get-context-keys-for-custom-policy 命令解析每个提供的策略，并列出这些策略使用的上下文键。使用此命令来确定必须提供哪些上下文键值才能成功使用策略模拟器

命令 simulate-custom-policy 和 simulate-custom-policy。您还可以使用 get-context-keys-for-custom-policy 命令检索与 IAM 用户或角色关联的所有策略使用的上下文键列表。以 file:// 开头的参数值指示命令读取文件的内容，然后使用内容而不是文件名本身作为参数的值。

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/${aws:username}","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
```

输出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

示例 2：列出作为文件输入提供的一个或多个自定义 JSON 策略所引用的上下文键

以下 get-context-keys-for-custom-policy 命令与前面的示例相同，只是策略是在文件中提供而不是作为参数提供。由于该命令需要 JSON 字符串列表而不是 JSON 结构列表，因此尽管您可以将其折叠成一个，但该文件的结构必须如下所示。

```
[
    "Policy1",
    "Policy2"
]
```

例如，包含上一个示例中策略的文件必须如下所示。您必须在策略字符串中每个嵌入的双引号前面加上反斜杠 " 来对其进行转义。

```
[ "{\"Version\": \"2012-10-17\", \"Statement\": {\"Effect\": \"Allow
\", \"Action\": \"dynamodb:*\", \"Resource\": \"arn:aws:dynamodb:us-
west-2:128716708097:table/${aws:username}\", \"Condition\": {\"DateGreaterThan\":
 {\"aws:CurrentTime\": \"2015-08-16T12:00:00Z\"}}}}" ]
```

然后，可以将此文件提交给以下命令。

```
aws iam get-context-keys-for-custom-policy \
    --policy-input-list file://policyfile.json
```

输出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用 IAM Policy Simulator（Amazon CLI 和 Amazon API）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetContextKeysForCustomPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例获取所提供策略 json 中存在的所有上下文键。为提供多个策略，您可以用逗号分隔值列表提供。

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetContextKeysForCustomPolicy。

Tools for PowerShell V5

示例 1：此示例获取所提供策略 json 中存在的所有上下文键。为提供多个策略，您可以用逗号分隔值列表提供。

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForCustomPolicy -PolicyInputList $policy1,$policy2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetContextKeysForCustomPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetContextKeysForPrincipalPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 GetContextKeysForPrincipalPolicy。

CLI

Amazon CLI

列出与 IAM 主体关联的所有策略引用的上下文键

以下 get-context-keys-for-principal-policy 命令检索附加到用户 saanvi 及其所属任何组的所有策略。然后，该命令会解析每个策略并列出这些策略使用的上下文键。使用此命令来确定必须提供哪些上下文键值才能成功使用 simulate-custom-policy 和 simulate-principal-policy 命令。您还可以使用 get-context-keys-for-custom-policy 命令检索任意 JSON 策略使用的上下文键列表。

```
aws iam get-context-keys-for-principal-policy \
   --policy-source-arn arn:aws:iam::123456789012:user/saanvi
```

输出：

```
{
    "ContextKeyNames": [
        "aws:username",
        "aws:CurrentTime"
    ]
```

```
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用 IAM Policy Simulator（Amazon CLI 和 Amazon API）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetContextKeysForPrincipalPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例获取所提供的策略 json 中和附加到 IAM 实体（用户/角色等）的策略中存在的所有上下文键。对于 -PolicyInputList，您可用逗号分隔值的形式提供多个值列表。

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetContextKeysForPrincipalPolicy。

Tools for PowerShell V5

示例 1：此示例获取所提供的策略 json 中和附加到 IAM 实体（用户/角色等）的策略中存在的所有上下文键。对于 -PolicyInputList，您可用逗号分隔值的形式提供多个值列表。

```
$policy1 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/","Condition":{"DateGreaterThan":
{"aws:CurrentTime":"2015-08-16T12:00:00Z"}}}}'
$policy2 = '{"Version":"2012-10-17",       "Statement":
{"Effect":"Allow","Action":"dynamodb:*","Resource":"arn:aws:dynamodb:us-
west-2:123456789012:table/"}}'
Get-IAMContextKeysForPrincipalPolicy -PolicyInputList $policy1,$policy2 -
PolicySourceArn arn:aws:iam::852640994763:user/TestUser
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetContextKeysForPrincipalPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetCredentialReport** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetCredentialReport。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

CLI

Amazon CLI

获取凭证报告

此示例打开返回的报告，并将其作为文本行数组输出到管道。

```
aws iam get-credential-report
```

输出：

```
{
    "GeneratedTime":  "2015-06-17T19:11:50Z",
    "ReportFormat": "text/csv"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的获取 Amazon 账户的凭证报告。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetCredentialReport。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例打开返回的报告，并将其作为文本行数组输出到管道。第一行是标题，含有用逗号分隔的列名。接下来的每一行都是一个用户的详细信息行，其中每个字段用逗号分隔。必须先使用 **Request-IAMCredentialReport** cmdlet 生成报告，然后才能查看报告。要将报告作为单个字符串检索，请使用 **-Raw** 而不是 **-AsTextArray**。**-AsTextArray** 开关也接受别名 **-SplitLines**。有关输出中列的完整列表，请参阅服务 API 参考。请注意，如果不使用 **-AsTextArray** 或 **-SplitLines**，则必须使用 .NET **StreamReader** 类从 **.Content** 属性中提取文本。

```
Request-IAMCredentialReport
```

输出：

```
Description                                                    State
-----------                                                    -----
No report exists. Starting a new report generation task        STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

输出：

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetCredentialReport。

Tools for PowerShell V5

示例 1：此示例打开返回的报告，并将其作为文本行数组输出到管道。第一行是标题，含有用逗号分隔的列名。接下来的每一行都是一个用户的详细信息行，其中每个字段用逗号分隔。必须先使用 **Request-IAMCredentialReport** cmdlet 生成报告，然后才能查看报告。要将报告作为单个字符串检索，请使用 **-Raw** 而不是 **-AsTextArray**。**-AsTextArray** 开关也接受别名 **-SplitLines**。有关输出中列的完整列表，请参阅服务 API 参考。请注意，如果不使用 **-AsTextArray** 或 **-SplitLines**，则必须使用 .NET **StreamReader** 类从 **.Content** 属性中提取文本。

```
Request-IAMCredentialReport
```

输出：

```
Description                                                          State
-----------                                                          -----
No report exists. Starting a new report generation task              STARTED
```

```
Get-IAMCredentialReport -AsTextArray
```

输出：

```
 user,arn,user_creation_time,password_enabled,password_last_used,password_last_changed,pa
 root_account,arn:aws:iam::123456789012:root,2014-10-15T16:31:25+00:00,not_supported,2015
A,false,N/A,false,N/A,false,N/A
Administrator,arn:aws:iam::123456789012:user/
Administrator,2014-10-16T16:03:09+00:00,true,2015-04-20T15:18:32+00:00,2014-10-16T16:06:0
A,false,true,2014-12-03T18:53:41+00:00,true,2015-03-25T20:38:14+00:00,false,N/
A,false,N/A
Bill,arn:aws:iam::123456789012:user/Bill,2015-04-15T18:27:44+00:00,false,N/A,N/
A,N/A,false,false,N/A,false,N/A,false,2015-04-20T20:00:12+00:00,false,N/A
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetCredentialReport。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetCredentialReport。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->getcredentialreport( ).
        MESSAGE 'Retrieved credential report.' TYPE 'I'.
      CATCH /aws1/cx_iamcredrptnotpresen00.
        MESSAGE 'Credential report not present.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptexpiredex.
        MESSAGE 'Credential report expired.' TYPE 'E'.
      CATCH /aws1/cx_iamcredrptnotreadyex.
        MESSAGE 'Credential report not ready.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when getting credential report.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetCredentialReport。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetGroup** 与 CLI 配合使用

以下代码示例演示如何使用 GetGroup。

CLI

Amazon CLI

获取 IAM 组

此示例返回有关 IAM 组 Admins 的详细信息。

```
aws iam get-group \
    --group-name Admins
```

输出：

```
{
    "Group": {
        "Path": "/",
        "CreateDate": "2015-06-16T19:41:48Z",
        "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
```

```
        "Arn": "arn:aws:iam::123456789012:group/Admins",
        "GroupName": "Admins"
    },
    "Users": []
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 身份（用户、用户组和角色）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回有关 IAM 组 **Testers** 的详细信息，包括属于该组的所有 IAM 用户的集合。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

输出：

```
Group                                    IsTruncated           Marker
     Users
-----                                    -----------           ------
     -----
Amazon.IdentityManagement.Model.Group    False
     {Theresa, David}
```

```
$results.Group
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetGroup。

Tools for PowerShell V5

示例 1：此示例返回有关 IAM 组 **Testers** 的详细信息，包括属于该组的所有 IAM 用户的集合。

```
$results = Get-IAMGroup -GroupName "Testers"
$results
```

输出：

```
Group                                    IsTruncated           Marker
     Users
-----                                    -----------           ------
     -----
Amazon.IdentityManagement.Model.Group    False
     {Theresa, David}
```

```
$results.Group
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
```

```
GroupId    : 3RHNZZGQJ7QHMAEXAMPLE1
GroupName  : Testers
Path       : /
```

```
$results.Users
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/Theresa
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 1/1/0001 12:00:00 AM
Path             : /
UserId           : 4OSVDDJJTF4XEEXAMPLE2
UserName         : Theresa

Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE3
UserName         : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetGroupPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 GetGroupPolicy。

CLI

Amazon CLI

获取有关附加到 IAM 组的策略的信息

以下 get-group-policy 命令获取有关附加到名为 Test-Group 的组的指定策略的信息。

```
aws iam get-group-policy \
```

```
    --group-name Test-Group \
    --policy-name S3-ReadOnly-Policy
```

输出：

```
{
    "GroupName": "Test-Group",
    "PolicyDocument": {
        "Statement": [
            {
                "Action": [
                    "s3:Get*",
                    "s3:List*"
                ],
                "Resource": "*",
                "Effect": "Allow"
            }
        ]
    },
    "PolicyName": "S3-ReadOnly-Policy"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetGroupPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回有关组 **Testers** 的名为 **PowerUserAccess-Testers** 的嵌入式内联策略的详细信息。**PolicyDocument** 属性采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

输出：

```
GroupName     PolicyDocument
 PolicyName
```

```
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetGroupPolicy。

Tools for PowerShell V5

示例 1：此示例返回有关组 **Testers** 的名为 **PowerUserAccess-Testers** 的嵌入式内联策略的详细信息。**PolicyDocument** 属性采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMGroupPolicy -GroupName Testers -PolicyName PowerUserAccess-
Testers
$results
```

输出：

```
GroupName     PolicyDocument
 PolicyName
---------     --------------
 ----------
Testers       %7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%0A%20...
 PowerUserAccess-Testers
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetGroupPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetInstanceProfile** 与 CLI 配合使用

以下代码示例演示如何使用 GetInstanceProfile。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- Amazon MSK 入门

CLI

Amazon CLI

获取有关实例配置文件的信息

以下 get-instance-profile 命令可获取名为 ExampleInstanceProfile 的实例配置文件的信息。

```
aws iam get-instance-profile \
    --instance-profile-name ExampleInstanceProfile
```

输出：

```
{
    "InstanceProfile": {
        "InstanceProfileId": "AID2MAB8DPLSRHEXAMPLE",
        "Roles": [
            {
                "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                "RoleId": "AIDGPMS9RO4H3FEXAMPLE",
                "CreateDate": "2013-01-09T06:33:26Z",
                "RoleName": "Test-Role",
                "Path": "/",
                "Arn": "arn:aws:iam::336924118301:role/Test-Role"
            }
        ],
        "CreateDate": "2013-06-12T23:52:02Z",
        "InstanceProfileName": "ExampleInstanceProfile",
        "Path": "/",
        "Arn": "arn:aws:iam::336924118301:instance-profile/
ExampleInstanceProfile"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetInstanceProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 Amazon 账户中定义的名为 **ec2instancerole** 的实例配置文件的详细信息。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
```

```
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》》中的 GetInstanceProfile。

Tools for PowerShell V5

示例 1：此示例返回当前 Amazon 账户中定义的名为 **ec2instancerole** 的实例配置文件的详细信息。

```
Get-IAMInstanceProfile -InstanceProfileName ec2instancerole
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetInstanceProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetLoginProfile** 与 CLI 配合使用

以下代码示例演示如何使用 GetLoginProfile。

CLI

Amazon CLI

获取 IAM 用户的密码信息

以下 get-login-profile 命令可获取名为 Bob 的 IAM 用户的密码相关信息。

```
aws iam get-login-profile \
    --user-name Bob
```

输出：

```
{
    "LoginProfile": {
        "UserName": "Bob",
        "CreateDate": "2012-09-21T23:03:39Z"
    }
}
```

get-login-profile 命令可用于验证 IAM 用户是否有密码。如果没有为用户定义密码，则该命令将返回 NoSuchEntity 错误。

您无法使用此命令查看密码。如果密码丢失，则可以为用户重置密码（update-login-profile）。或者，您可以删除用户的登录配置文件（delete-login-profile），然后创建新的登录配置文件（create-login-profile）。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的密码。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetLoginProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回密码创建日期，以及 IAM 用户 **David** 是否需要重置密码。

```
Get-IAMLoginProfile -UserName David
```

输出：

```
CreateDate                PasswordResetRequired                UserName
----------                ---------------------                --------
12/10/2014 3:39:44 PM     False                                David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetLoginProfile。

Tools for PowerShell V5

示例 1：此示例返回密码创建日期，以及 IAM 用户 **David** 是否需要重置密码。

```
Get-IAMLoginProfile -UserName David
```

输出：

```
CreateDate                 PasswordResetRequired                UserName
----------                 ---------------------                --------
12/10/2014 3:39:44 PM      False                                David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetLoginProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetOpenIdConnectProvider** 与 CLI 配合使用

以下代码示例演示如何使用 GetOpenIdConnectProvider。

CLI

Amazon CLI

返回有关指定 OpenID Connect 提供者的信息

此示例返回有关 ARN 为 arn:aws:iam::123456789012:oidc-provider/server.example.com 的 OpenID Connect 提供者的详细信息。

```
aws iam get-open-id-connect-provider \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
server.example.com
```

输出：

```
{
    "Url": "server.example.com"
        "CreateDate": "2015-06-16T19:41:48Z",
```

```
        "ThumbprintList": [
        "12345abcdefghijk67890lmnopqrst987example"
        ],
        "ClientIDList": [
        "example-application-ID"
        ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetOpenIdConnectProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回有关 ARN 为 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** 的 OpenID Connect 提供者的详细信息。**ClientIDList** 属性是一个集合，其中包含为此提供者定义的所有客户端 ID。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

输出：

```
ClientIDList         CreateDate               ThumbprintList
           Url
------------         ----------               --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetOpenIdConnectProvider。

Tools for PowerShell V5

示例 1：此示例返回有关 ARN 为 **arn:aws:iam::123456789012:oidc-provider/accounts.google.com** 的 OpenID Connect 提供者的详细信息。**ClientIDList** 属性是一个集合，其中包含为此提供者定义的所有客户端 ID。

```
Get-IAMOpenIDConnectProvider -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/oidc.example.com
```

输出：

```
ClientIDList         CreateDate                ThumbprintList
           Url
------------         ----------                --------------
           ---
{MyOIDCApp}          2/3/2015 3:00:30 PM
 {12345abcdefghijk67890lmnopqrst98765uvwxy}   oidc.example.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetOpenIdConnectProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetPolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- Amazon MSK 入门
- 使用 IAM Policy Builder API

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
```

```
    /// Get information about an IAM policy.
    /// </summary>
    /// <param name="policyArn">The IAM policy to retrieve information for.</
param>
    /// <returns>The IAM policy.</returns>
    public async Task<ManagedPolicy> GetPolicyAsync(string policyArn)
    {

        var response = await _IAMService.GetPolicyAsync(new GetPolicyRequest
 { PolicyArn = policyArn });
        return response.Policy;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 GetPolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::getPolicy(const Aws::String &policyArn,
                            const Aws::Client::ClientConfiguration &clientConfig)
 {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetPolicyRequest request;
    request.SetPolicyArn(policyArn);

    auto outcome = iam.GetPolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error getting policy " << policyArn << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        const auto &policy = outcome.GetResult().GetPolicy();
        std::cout << "Name: " << policy.GetPolicyName() << std::endl <<
```

```
                  "ID: " << policy.GetPolicyId() << std::endl << "Arn: " <<
                  policy.GetArn() << std::endl << "Description: " <<
                  policy.GetDescription() << std::endl << "CreateDate: " <<

 policy.GetCreateDate().ToGmtString(Aws::Utils::DateFormat::ISO_8601)
                  << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 GetPolicy。

CLI

Amazon CLI

检索有关指定托管策略的信息

此示例将返回 ARN 为 `arn:aws:iam::123456789012:policy/MySamplePolicy` 的托管策略的详细信息。

```
aws iam get-policy \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
{
    "Policy": {
        "PolicyName": "MySamplePolicy",
        "CreateDate": "2015-06-17T19:23;32Z",
        "AttachmentCount": 0,
        "IsAttachable": true,
        "PolicyId": "Z27SI6FQMGNQ2EXAMPLE1",
        "DefaultVersionId": "v1",
        "Path": "/",
        "Arn": "arn:aws:iam::123456789012:policy/MySamplePolicy",
        "UpdateDate": "2015-06-17T19:23:32Z"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetPolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// GetPolicy gets data about a policy.
func (wrapper PolicyWrapper) GetPolicy(ctx context.Context, policyArn string)
 (*types.Policy, error) {
 var policy *types.Policy
 result, err := wrapper.IamClient.GetPolicy(ctx, &iam.GetPolicyInput{
  PolicyArn: aws.String(policyArn),
 })
```

```
 if err != nil {
  log.Printf("Couldn't get policy %v. Here's why: %v\n", policyArn, err)
 } else {
  policy = result.Policy
 }
 return policy, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 GetPolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

获取策略。

```
import { GetPolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} policyArn
 */
export const getPolicy = (policyArn) => {
  const command = new GetPolicyCommand({
    PolicyArn: policyArn,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-policies.html#iam-examples-policies-getting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetPolicy。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  PolicyArn: "arn:aws:iam::aws:policy/AWSLambdaExecute",
};

iam.getPolicy(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data.Policy.Description);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-policies.html#iam-examples-policies-getting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetPolicy。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun getIAMPolicy(policyArnVal: String?) {
    val request =
        GetPolicyRequest {
            policyArn = policyArnVal
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.getPolicy(request)
        println("Successfully retrieved policy ${response.policy?.policyName}")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 GetPolicy。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getPolicy($policyArn)
    {
```

```
        return $this->customWaiter(function () use ($policyArn) {
            return $this->iamClient->getPolicy(['PolicyArn' => $policyArn]);
        });
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 GetPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的托管策略的详细信息。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetPolicy。

Tools for PowerShell V5

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MySamplePolicy** 的托管策略的详细信息。

```
Get-IAMPolicy -PolicyArn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
Arn              : arn:aws:iam::aws:policy/MySamplePolicy
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : MySamplePolicy
UpdateDate       : 2/6/2015 10:40:08 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetPolicy。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
```

```
        raise
    else:
        return policy_statement
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Fetches an IAM policy by its ARN
 # @param policy_arn [String] the ARN of the IAM policy to retrieve
 # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
 def get_policy(policy_arn)
   response = @iam_client.get_policy(policy_arn: policy_arn)
   policy = response.policy
   @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
   policy
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
   raise
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
   raise
 end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 GetPolicy。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->getpolicy( iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetPolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func getPolicy(arn: String) async throws -> IAMClientTypes.Policy {
        let input = GetPolicyInput(
            policyArn: arn
        )
        do {
```

```
            let output = try await client.getPolicy(input: input)
            guard let policy = output.policy else {
                throw ServiceHandlerError.noSuchPolicy
            }
            return policy
        } catch {
            print("ERROR: getPolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 GetPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetPolicyVersion** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetPolicyVersion。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 管理策略
- 使用 IAM Policy Builder API

CLI

Amazon CLI

检索有关指定托管策略的指定版本的信息

此示例将返回 ARN 为 arn:aws:iam::123456789012:policy/MyManagedPolicy 的策略 v2 版本的策略文档。

```
aws iam get-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

输出：

```
{
    "PolicyVersion": {
        "Document": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Action": "iam:*",
                    "Resource": "*"
                }
            ]
        },
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2023-04-11T00:22:54+00:00"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetPolicyVersion。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略 **v2** 版本的策略文档。**Document** 属性中的策略文档采用 URL 编码，在本示例中使用 **UrlDecode** .NET 方法进行解码。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

输出：

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 -----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetPolicyVersion。

Tools for PowerShell V5

示例 1：此示例将返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略 **v2** 版本的策略文档。**Document** 属性中的策略文档采用 URL 编码，在本示例中使用 **UrlDecode** .NET 方法进行解码。

```
$results = Get-IAMPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy -VersionId v2
$results
```

输出：

```
CreateDate             Document
 IsDefaultVersion     VersionId
----------             --------
 ----------------     ---------
2/12/2015 9:39:53 AM   %7B%0A%20%20%22Version%22%3A%20%222012-10...    True
           v2

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
$policy = [System.Web.HttpUtility]::UrlDecode($results.Document)
$policy
```

```
{
  "Version":"2012-10-17",
  "Statement":
    {
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeInstances"
      ],
      "Resource": [
        "arn:aws:ec2:us-east-1:555555555555:instance/i-b188560f"
      ]
    }
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetPolicyVersion。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
```

```
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
policy_arn)
        raise
    else:
        return policy_statement
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetPolicyVersion。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->getpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Retrieved policy version information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'Invalid input provided.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetPolicyVersion。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetRole** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetRole。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 配置 Amazon ECS Service Connect
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 创建 EC2 启动类型的 Amazon ECS 服务
- Redshift Serverless 入门
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon Redshift 预置集群入门
- Amazon SageMaker Feature Store 入门
- IoT Device Defender 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
/// <summary>
/// Get information about an IAM role.
/// </summary>
/// <param name="roleName">The name of the IAM role to retrieve information
/// for.</param>
/// <returns>The IAM role that was retrieved.</returns>
public async Task<Role> GetRoleAsync(string roleName)
{
    var response = await _IAMService.GetRoleAsync(new GetRoleRequest
    {
```

```
            RoleName = roleName,
        });

        return response.Role;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 GetRole。

CLI

Amazon CLI

获取有关 IAM 角色的信息

以下 `get-role` 命令可获取名为 `Test-Role` 的角色的信息。

```
aws iam get-role \
    --role-name Test-Role
```

输出：

```
{
    "Role": {
        "Description": "Test Role",
        "AssumeRolePolicyDocument":"<URL-encoded-JSON>",
        "MaxSessionDuration": 3600,
        "RoleId": "AROA1234567890EXAMPLE",
        "CreateDate": "2019-11-13T16:45:56Z",
        "RoleName": "Test-Role",
        "Path": "/",
        "RoleLastUsed": {
            "Region": "us-east-1",
            "LastUsedDate": "2019-11-13T17:14:00Z"
        },
        "Arn": "arn:aws:iam::123456789012:role/Test-Role"
    }
}
```

该命令会显示附加到角色的信任策略。要列出附加到角色的权限策略，请使用 `list-role-policies` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetRole。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// GetRole gets data about a role.
func (wrapper RoleWrapper) GetRole(ctx context.Context, roleName string)
 (*types.Role, error) {
 var role *types.Role
 result, err := wrapper.IamClient.GetRole(ctx,
  &iam.GetRoleInput{RoleName: aws.String(roleName)})
 if err != nil {
  log.Printf("Couldn't get role %v. Here's why: %v\n", roleName, err)
```

```
 } else {
  role = result.Role
 }
 return role, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 GetRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

获取角色。

```
import { GetRoleCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 */
export const getRole = (roleName) => {
  const command = new GetRoleCommand({
    RoleName: roleName,
  });

  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetRole。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function getRole($roleName)
    {
        return $this->customWaiter(function () use ($roleName) {
            return $this->iamClient->getRole(['RoleName' => $roleName]);
        });
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 GetRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回 **lamda_exec_role** 的详细信息。其包括指定谁可以担任此角色的信任策略文档。策略文档采用 URL 编码，可使用 .NET **UrlDecode** 方法进行解码。在此示例中，原始策略在上传到策略之前已删除所有空格。要查看确定承担该角色的人员可以执行哪些操作的权限策略文档，对内联策略使用 **Get-IAMRolePolicy**，对附加的托管策略使用 **Get-IAMPolicyVersion**。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

输出：

```
Arn                   : arn:aws:iam::123456789012:role/lambda_exec_role
```

```
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
                           %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                           %22%7D%5D%7D
CreateDate               : 4/2/2015 9:16:11 AM
Path                     : /
RoleId                   : 2YBIKAIBHNKB4EXAMPLE1
RoleName                 : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

输出：

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetRole。

Tools for PowerShell V5

示例 1：此示例返回 **lamda_exec_role** 的详细信息。其包括指定谁可以担任此角色的信任策略文档。策略文档采用 URL 编码，可使用 .NET **UrlDecode** 方法进行解码。在此示例中，原始策略在上传到策略之前已删除所有空格。要查看确定承担该角色的人员可以执行哪些操作的权限策略文档，对内联策略使用 **Get-IAMRolePolicy**，对附加的托管策略使用 **Get-IAMPolicyVersion**。

```
$results = Get-IamRole -RoleName lambda_exec_role
$results | Format-List
```

输出：

```
Arn                      : arn:aws:iam::123456789012:role/lambda_exec_role
AssumeRolePolicyDocument : %7B%22Version%22%3A%222012-10-17%22%2C%22Statement
%22%3A%5B%7B%22Sid%22
                           %3A%22%22%2C%22Effect%22%3A%22Allow%22%2C%22Principal
%22%3A%7B%22Service
```

```
                            %22%3A%22lambda.amazonaws.com%22%7D%2C%22Action%22%3A
%22sts%3AAssumeRole
                            %22%7D%5D%7D
CreateDate                : 4/2/2015 9:16:11 AM
Path                      : /
RoleId                    : 2YBIKAIBHNKB4EXAMPLE1
RoleName                  : lambda_exec_role
```

```
$policy = [System.Web.HttpUtility]::UrlDecode($results.AssumeRolePolicyDocument)
$policy
```

输出：

```
{"Version":"2012-10-17",       "Statement":
[{"Sid":"","Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def get_role(role_name):
    """
    Gets a role by name.

    :param role_name: The name of the role to retrieve.
    :return: The specified role.
    """
    try:
        role = iam.Role(role_name)
```

```
        role.load()  # calls GetRole to load attributes
        logger.info("Got role with arn %s.", role.arn)
    except ClientError:
        logger.exception("Couldn't get role named %s.", role_name)
        raise
    else:
        return role
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 GetRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Gets data about a role.
#
# @param name [String] The name of the role to look up.
# @return [Aws::IAM::Role] The retrieved role.
def get_role(name)
  role = @iam_client.get_role({
                               role_name: name
                             }).role
  puts("Got data for role '#{role.role_name}'. Its ARN is '#{role.arn}'.")
rescue Aws::Errors::ServiceError => e
  puts("Couldn't get data for role '#{name}' Here's why:")
  puts("\t#{e.code}: #{e.message}")
  raise
else
  role
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 GetRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn get_role(
    client: &iamClient,
    role_name: String,
) -> Result<GetRoleOutput, SdkError<GetRoleError>> {
    let response = client.get_role().role_name(role_name).send().await?;
    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 GetRole。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->getrole( iv_rolename = iv_role_name ).
        MESSAGE 'Retrieved role information.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Role does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 GetRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func getRole(name: String) async throws -> IAMClientTypes.Role {
        let input = GetRoleInput(
            roleName: name
        )
        do {
            let output = try await client.getRole(input: input)
            guard let role = output.role else {
                throw ServiceHandlerError.noSuchRole
            }
            return role
        } catch {
            print("ERROR: getRole:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 GetRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetRolePolicy** 与 CLI 配合使用

以下代码示例演示如何使用 GetRolePolicy。

CLI

Amazon CLI

获取有关附加到 IAM 角色的策略的信息

以下 get-role-policy 命令获取有关附加到名为 Test-Role 的角色的指定策略的信息。

```
aws iam get-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy
```

输出：

```
{
  "RoleName": "Test-Role",
  "PolicyDocument": {
      "Statement": [
          {
              "Action": [
                  "s3:ListBucket",
                  "s3:Put*",
                  "s3:Get*",
                  "s3:*MultipartUpload*"
              ],
              "Resource": "*",
              "Effect": "Allow",
              "Sid": "1"
          }
      ]
  }
  "PolicyName": "ExamplePolicy"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetRolePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回嵌入在 IAM 角色 **lamda_exec_role** 中的名为 **oneClick_lambda_exec_role_policy** 的策略的权限策略文档。生成的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

输出：

```
PolicyDocument                                                PolicyName
              UserName
--------------                                                ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

输出：

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
```

```
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetRolePolicy。

Tools for PowerShell V5

示例 1：此示例返回嵌入在 IAM 角色 **lamda_exec_role** 中的名为 **oneClick_lambda_exec_role_policy** 的策略的权限策略文档。生成的策略文档采用 URL 编码。在本例中，使用 **UrlDecode** .NET 方法对其进行解码。

```
$results = Get-IAMRolePolicy -RoleName lambda_exec_role -PolicyName
 oneClick_lambda_exec_role_policy
$results
```

输出：

```
PolicyDocument                                                PolicyName
              UserName
--------------                                                ----------
              --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...
 oneClick_lambda_exec_role_policy     lambda_exec_role
```

```
[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
```

输出：

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
```

```
      "Action": [
        "logs:*"
      ],
      "Resource": "arn:aws:logs:us-east-1:555555555555:log-group:/aws/lambda/aws-
example-function:*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetSamlProvider** 与 CLI 配合使用

以下代码示例演示如何使用 GetSamlProvider。

CLI

Amazon CLI

检索 SAML 提供者元文档

此示例检索有关 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 SAML 2.0 提供者的详细信息。响应包括您从身份提供者那里获得的、用于创建 Amazon SAML 提供者实体的元数据文档，以及创建日期和到期日期。

```
aws iam get-saml-provider \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

输出：

```
{
    "SAMLMetadataDocument": "...SAMLMetadataDocument-XML...",
    "CreateDate": "2017-03-06T22:29:46+00:00",
    "ValidUntil": "2117-03-06T22:29:46.433000+00:00",
    "Tags": [
        {
            "Key": "DeptID",
            "Value": "123456"
        },
        {
            "Key": "Department",
            "Value": "Accounting"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM SAML 身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetSamlProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索有关 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 SAML 2.0 提供者的详细信息。响应包括您从身份提供者那里获得的、用于创建 Amazon SAML 提供者实体的元数据文档，以及创建日期和到期日期。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

输出：

```
CreateDate                 SAMLMetadataDocument
        ValidUntil
----------                 --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetSamlProvider。

Tools for PowerShell V5

示例 1：此示例检索有关 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 SAML 2.0 提供者的详细信息。响应包括您从身份提供者那里获得的、用于创建 Amazon SAML 提供者实体的元数据文档，以及创建日期和到期日期。

```
Get-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS
```

输出：

```
CreateDate                 SAMLMetadataDocument
        ValidUntil
----------                 --------------------
        ----------
12/23/2014 12:16:55 PM    <EntityDescriptor ID="_12345678-1234-5678-9012-
example1...    12/23/2114 12:16:54 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetSamlProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetServerCertificate** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetServerCertificate。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::getServerCertificate(const Aws::String &certificateName,
                                       const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::GetServerCertificateRequest request;
    request.SetServerCertificateName(certificateName);

    auto outcome = iam.GetServerCertificate(request);
    bool result = true;
    if (!outcome.IsSuccess()) {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error getting server certificate " << certificateName
 <<
                      ": " << outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << certificateName
                      << "' not found." << std::endl;
        }
    }
    else {
        const auto &certificate = outcome.GetResult().GetServerCertificate();
        std::cout << "Name: " <<

 certificate.GetServerCertificateMetadata().GetServerCertificateName()
                  << std::endl << "Body: " << certificate.GetCertificateBody() <<
                  std::endl << "Chain: " << certificate.GetCertificateChain() <<
                  std::endl;
    }

    return result;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 GetServerCertificate。

CLI

Amazon CLI

获取有关 Amazon 账户中服务器证书的详细信息

以下 `get-server-certificate` 命令将检索 Amazon 账户中指定服务器证书的所有详细信息。

```
aws iam get-server-certificate \
    --server-certificate-name myUpdatedServerCertificate
```

输出：

```
{
    "ServerCertificate": {
        "ServerCertificateMetadata": {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        "CertificateBody": "-----BEGIN CERTIFICATE-----
            MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
            VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6
            b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd
            BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN
            MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD
            VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z
            b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
            YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
            21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
            rDHudUZg3qX4waLG5M43q7Wgc/MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE
            Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
            nUhVVxYUntneD9+h8Mg9q6q+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
            FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
            NYiytVbZPQUQ5Yaxu2jXnimvrszlaEXAMPLE=-----END CERTIFICATE-----",
        "CertificateChain": "-----BEGIN CERTIFICATE-----\nMIICiTCCAfICCQD6md
            7oRw0uXOjANBgkqhkiG9w0BAqQUFADCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgT
            AldBMRAwDgYDVQQHEwdTZWF0drGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAs
```

```
            TC0lBTSBDb25zb2xlMRIwEAYDVsQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQ
            jb20wHhcNMTEwNDI1MjA0NTIxWhtcNMTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBh
            MCVVMxCzAJBgNVBAgTAldBMRAwDgsYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBb
            WF6b24xFDASBgNVBAsTC0lBTSBDb2d5zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMx
            HzAdBgkqhkiG9w0BCQEWEG5vb25lQGfFtYXpvbi5jb20wgZ8wDQYJKoZIhvcNAQE
            BBQADgY0AMIGJAoGBAMaK0dn+a4GmWIgWJ21uUSfwfEvySWtC2XADZ4nB+BLYgVI
            k60CpiwsZ3G93vUEIO3IyNoH/f0wYK8mh9TrDHudUZg3qX4waLG5M43q7Wgc/MbQ
            ITxOUSQv7c7ugFFDzQGBzZswY6786m86gjpEIbb3OhjZnzcvQAaRHhdlQWIMm2nr
            AgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCku4nUhVVxYUntneD9+h8Mg9q6q+auN
            KyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0FlkbFFBjvSfpJIlJ00zbhNYS5f6Guo
            EDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjS;TbNYiytVbZPQUQ5Yaxu2jXnimvw
            3rrszlaEWEG5vb25lQGFtsYXpvbiEXAMPLE=\n-----END CERTIFICATE-----"
    }
}
```

要列出 Amazon 账户中可用的服务器证书，请使用 `list-server-certificates` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 中管理服务器证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetServerCertificate。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

获取服务器证书。

```
import { GetServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} certName
 * @returns
 */
```

```
export const getServerCertificate = async (certName) => {
  const command = new GetServerCertificateCommand({
    ServerCertificateName: certName,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-getting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetServerCertificate。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.getServerCertificate(
  { ServerCertificateName: "CERTIFICATE_NAME" },
  function (err, data) {
    if (err) {
      console.log("Error", err);
    } else {
      console.log("Success", data);
    }
  }
```

```
);
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-getting。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 GetServerCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索有关名为 **MyServerCertificate** 的服务器证书的详细信息。您可以在 **CertificateBody** 和 **ServerCertificateMetadata** 属性中找到证书详细信息。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

输出：

```
CertificateBody          : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                              21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
```

```
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetServerCertificate。

Tools for PowerShell V5

示例 1：此示例检索有关名为 **MyServerCertificate** 的服务器证书的详细信息。您可以在 **CertificateBody** 和 **ServerCertificateMetadata** 属性中找到证书详细信息。

```
$result = Get-IAMServerCertificate -ServerCertificateName MyServerCertificate
$result | format-list
```

输出：

```
CertificateBody           : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC
```

```
 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt

 YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn+a4GmWIWJ
                            21uUSfwfEvySWtC2XADZ4nB
+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/f0wYK8m9T
                            rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                            nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                            NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                            -----END CERTIFICATE-----
CertificateChain          :
ServerCertificateMetadata :
 Amazon.IdentityManagement.Model.ServerCertificateMetadata
```

```
$result.ServerCertificateMetadata
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetServerCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetServiceLastAccessedDetails** 与 CLI 配合使用

以下代码示例演示如何使用 GetServiceLastAccessedDetails。

CLI

Amazon CLI

检索服务访问报告

以下 get-service-last-accessed-details 示例检索之前生成的报告，其中列出了 IAM 实体所访问的服务。要生成报告，请使用 generate-service-last-accessed-details 命令。

```
aws iam get-service-last-accessed-details \
    --job-id 2eb6c2b8-7b4c-3xmp-3c13-03b72c8cdfdc
```

输出：

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:50:35.929Z",
    "ServicesLastAccessed": [
        ...
        {
            "ServiceName": "AWS Lambda",
            "LastAuthenticated": "2019-09-30T23:02:00Z",
            "ServiceNamespace": "lambda",
            "LastAuthenticatedEntity": "arn:aws:iam::123456789012:user/admin",
            "TotalAuthenticatedEntities": 6
        },
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用上次访问的信息优化 Amazon 中的权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetServiceLastAccessedDetails。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例提供了请求调用中关联的 IAM 实体（用户、组、角色或策略）上次访问的服务的详细信息。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

输出：

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetServiceLastAccessedDetails。

Tools for PowerShell V5

示例 1：此示例提供了请求调用中关联的 IAM 实体（用户、组、角色或策略）上次访问的服务的详细信息。

```
Request-IAMServiceLastAccessedDetail -Arn arn:aws:iam::123456789012:user/TestUser
```

输出：

```
f0b7a819-eab0-929b-dc26-ca598911cb9f
```

```
Get-IAMServiceLastAccessedDetail -JobId f0b7a819-eab0-929b-dc26-ca598911cb9f
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetServiceLastAccessedDetails。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetServiceLastAccessedDetailsWithEntities** 与 CLI 配合使用

以下代码示例演示如何使用 GetServiceLastAccessedDetailsWithEntities。

CLI

Amazon CLI

检索包含服务详细信息的服务访问报告

以下 get-service-last-accessed-details-with-entities 示例检索一份报告，其中包含有关访问指定服务的 IAM 用户和其他实体的详细信息。要生成报告，请使用 generate-service-last-accessed-details 命令。要获取使用命名空间访问的服务列表，请使用 get-service-last-accessed-details。

```
aws iam get-service-last-accessed-details-with-entities \
    --job-id 78b6c2ba-d09e-6xmp-7039-ecde30b26916 \
    --service-namespace lambda
```

输出：

```
{
    "JobStatus": "COMPLETED",
    "JobCreationDate": "2019-10-01T03:55:41.756Z",
    "JobCompletionDate": "2019-10-01T03:55:42.533Z",
    "EntityDetailsList": [
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/admin",
                "Name": "admin",
                "Type": "USER",
                "Id": "AIDAIO2XMPLENQEXAMPLE",
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-30T23:02:00Z"
        },
        {
            "EntityInfo": {
                "Arn": "arn:aws:iam::123456789012:user/developer",
                "Name": "developer",
                "Type": "USER",
                "Id": "AIDAIBEYXMPL2YEXAMPLE",
```

```
                "Path": "/"
            },
            "LastAuthenticated": "2019-09-16T19:34:00Z"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用上次访问的信息优化 Amazon 中的权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetServiceLastAccessedDetailsWithEntities。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例提供了相应的 IAM 实体上次访问请求中服务的时间戳。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

输出：

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

输出：

```
EntityInfo                                    LastAuthenticated
----------                                    -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

输出：

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetServiceLastAccessedDetailsWithEntities。

Tools for PowerShell V5

示例 1：此示例提供了相应的 IAM 实体上次访问请求中服务的时间戳。

```
$results = Get-IAMServiceLastAccessedDetailWithEntity -JobId f0b7a819-eab0-929b-
dc26-ca598911cb9f -ServiceNamespace ec2
$results
```

输出：

```
EntityDetailsList : {Amazon.IdentityManagement.Model.EntityDetails}
Error             :
IsTruncated       : False
JobCompletionDate : 12/29/19 11:19:31 AM
JobCreationDate   : 12/29/19 11:19:31 AM
JobStatus         : COMPLETED
Marker            :
```

```
$results.EntityDetailsList
```

输出：

```
EntityInfo                                  LastAuthenticated
----------                                  -----------------
Amazon.IdentityManagement.Model.EntityInfo 11/16/19 3:47:00 PM
```

```
$results.EntityInfo
```

输出：

```
Arn  : arn:aws:iam::123456789012:user/TestUser
Id   : AIDA4NBK5CXF5TZHU1234
Name : TestUser
Path : /
Type : USER
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetServiceLastAccessedDetailsWithEntities。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetServiceLinkedRoleDeletionStatus** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetServiceLinkedRoleDeletionStatus。

CLI

Amazon CLI

查看删除服务相关角色的请求状态

以下 get-service-linked-role-deletion-status 示例演示了先前请求删除服务相关角色的状态。删除操作异步进行。当您发出请求时，您将得到一个 DeletionTaskId 值，该值将作为此命令的参数提供。

```
aws iam get-service-linked-role-deletion-status \
    --deletion-task-id task/aws-service-role/lex.amazonaws.com/
AWSServiceRoleForLexBots/1a2b3c4d-1234-abcd-7890-abcdeEXAMPLE
```

输出：

```
{
"Status": "SUCCEEDED"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用服务相关角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetServiceLinkedRoleDeletionStatus。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import {
  GetServiceLinkedRoleDeletionStatusCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} deletionTaskId
 */
export const getServiceLinkedRoleDeletionStatus = (deletionTaskId) => {
  const command = new GetServiceLinkedRoleDeletionStatusCommand({
    DeletionTaskId: deletionTaskId,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的 GetServiceLinkedRoleDeletionStatus。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetUser** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetUser。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Get information about an IAM user.
    /// </summary>
    /// <param name="userName">The username of the user.</param>
    /// <returns>An IAM user object.</returns>
    public async Task<User> GetUserAsync(string userName)
    {
        var response = await _IAMService.GetUserAsync(new GetUserRequest
{ UserName = userName });
        return response.User;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 GetUser。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
```

```
# This function outputs everything sent to it to STDERR (standard error output).
#############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

#############################################################################
# function iam_user_exists
#
# This function checks to see if the specified AWS Identity and Access Management
 (IAM) user already exists.
#
# Parameters:
#       $1 - The name of the IAM user to check.
#
# Returns:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
#############################################################################
function iam_user_exists() {
  local user_name
  user_name=$1

  # Check whether the IAM user already exists.
  # We suppress all output - we're interested only in the return code.

  local errors
  errors=$(aws iam get-user \
    --user-name "$user_name" 2>&1 >/dev/null)

  local error_code=${?}

  if [[ $error_code -eq 0 ]]; then
    return 0 # 0 in Bash script means true.
  else
    if [[ $errors != *"error"*"(NoSuchEntity)"* ]]; then
      aws_cli_error_log $error_code
      errecho "Error calling iam get-user $errors"
    fi

    return 1 # 1 in Bash script means false.
  fi
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetUser。

CLI

Amazon CLI

获取有关 IAM 用户的信息

以下 `get-user` 命令可获取名为 `Paulo` 的 IAM 用户的信息。

```
aws iam get-user \
    --user-name Paulo
```

输出：

```
{
    "User": {
        "UserName": "Paulo",
        "Path": "/",
        "CreateDate": "2019-09-21T23:03:13Z",
        "UserId": "AIDA123456789EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:user/Paulo"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetUser。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// GetUser gets data about a user.
func (wrapper UserWrapper) GetUser(ctx context.Context, userName string)
 (*types.User, error) {
 var user *types.User
 result, err := wrapper.IamClient.GetUser(ctx, &iam.GetUserInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  var apiError smithy.APIError
  if errors.As(err, &apiError) {
   switch apiError.(type) {
   case *types.NoSuchEntityException:
    log.Printf("User %v does not exist.\n", userName)
    err = nil
   default:
    log.Printf("Couldn't get user %v. Here's why: %v\n", userName, err)
   }
  }
 } else {
  user = result.User
 }
 return user, err
```

```
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 GetUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索名为 **David** 的用户的详细信息。

```
Get-IAMUser -UserName David
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

示例 2：此示例检索有关当前登录的 IAM 用户的详细信息。

```
Get-IAMUser
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetUser。

Tools for PowerShell V5

示例 1：此示例检索名为 **David** 的用户的详细信息。

```
Get-IAMUser -UserName David
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/David
CreateDate       : 12/10/2014 3:39:27 PM
PasswordLastUsed : 3/19/2015 8:44:04 AM
Path             : /
UserId           : Y4FKWQCXTA52QEXAMPLE1
UserName         : David
```

示例 2：此示例检索有关当前登录的 IAM 用户的详细信息。

```
Get-IAMUser
```

输出：

```
Arn              : arn:aws:iam::123456789012:user/Bob
CreateDate       : 10/16/2014 9:03:09 AM
PasswordLastUsed : 3/4/2015 12:12:33 PM
Path             : /
UserId           : 7K3GJEANSKZF2EXAMPLE2
UserName         : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetUser。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Retrieves a user's details
#
```

```
  # @param user_name [String] The name of the user to retrieve
  # @return [Aws::IAM::Types::User, nil] The user object if found, or nil if an
error occurred
 def get_user(user_name)
   response = @iam_client.get_user(user_name: user_name)
   response.user
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("User '#{user_name}' not found.")
   nil
 rescue Aws::IAM::Errors::ServiceError => e
   @logger.error("Error retrieving user '#{user_name}': #{e.message}")
   nil
 end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 GetUser。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func getUser(name: String? = nil) async throws -> IAMClientTypes.User
{
        let input = GetUserInput(
            userName: name
        )
        do {
            let output = try await iamClient.getUser(input: input)
            guard let user = output.user else {
                throw ServiceHandlerError.noSuchUser
```

```
            }
            return user
        } catch {
            print("ERROR: getUser:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API Reference》中的 GetUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetUserPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 GetUserPolicy。

CLI

Amazon CLI

列出 IAM 用户的策略详细信息

以下 get-user-policy 命令将列出附加到名为 Bob 的 IAM 用户的指定策略的详细信息。

```
aws iam get-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy
```

输出：

```
{
    "UserName": "Bob",
    "PolicyName": "ExamplePolicy",
    "PolicyDocument": {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Action": "*",
                "Resource": "*",
                "Effect": "Allow"
```

```
            }
        ]
    }
}
```

要获取 IAM 用户的策略列表，请使用 `list-user-policies` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetUserPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中名为 **Davids_IAM_Admin_Policy** 的内联策略的详细信息。策略文档采用 URL 编码。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

输出：

```
PolicyDocument                                                  PolicyName
       UserName
--------------                                                  ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
```

```
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetUserPolicy。

Tools for PowerShell V5

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中名为 **Davids_IAM_Admin_Policy** 的内联策略的详细信息。策略文档采用 URL 编码。

```
$results = Get-IAMUserPolicy -PolicyName Davids_IAM_Admin_Policy -UserName David
$results
```

输出：

```
PolicyDocument                                                  PolicyName
       UserName
--------------                                                  ----------
       --------
%7B%0A%20%20%22Version%22%3A%20%222012-10-17%22%2C%...    Davids_IAM_Admin_Policy
       David

[System.Reflection.Assembly]::LoadWithPartialName("System.Web.HttpUtility")
[System.Web.HttpUtility]::UrlDecode($results.PolicyDocument)
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:GetUser",
        "iam:ListUsers"
      ],
      "Resource": [
        "arn:aws:iam::111122223333:user/*"
      ]
    }
  ]
}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetUserPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListAccessKeys** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListAccessKeys。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问密钥

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_list_access_keys
#
# This function lists the access keys for the specified user.
#
# Parameters:
#       -u user_name -- The name of the IAM user.
```

```
#
# Returns:
#       access_key_ids
#     And:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_list_access_keys() {

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_access_keys"
    echo "Lists the AWS Identity and Access Management (IAM) access key IDs for
 the specified user."
    echo "  -u user_name   The name of the IAM user."
    echo ""
  }

  local user_name response
  local option OPTARG # Required to use getopts command in a function.
  # Retrieve the calling parameters.
  while getopts "u:h" option; do
    case "${option}" in
      u) user_name="${OPTARG}" ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  if [[ -z "$user_name" ]]; then
    errecho "ERROR: You must provide a username with the -u parameter."
    usage
    return 1
  fi

  response=$(aws iam list-access-keys \
```

```
    --user-name "$user_name" \
    --output text \
    --query 'AccessKeyMetadata[].AccessKeyId')

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-access-keys operation failed.$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 ListAccessKeys。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::listAccessKeys(const Aws::String &userName,
                                 const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccessKeysRequest request;
    request.SetUserName(userName);

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccessKeys(request);
        if (!outcome.IsSuccess()) {
```

```
            std::cerr << "Failed to list access keys for user " << userName
                      << ": " << outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "UserName" <<
                      std::setw(30) << "KeyID" << std::setw(20) << "Status" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }

        const auto &keys = outcome.GetResult().GetAccessKeyMetadata();
        const Aws::String DATE_FORMAT = "%Y-%m-%d";

        for (const auto &key: keys) {
            Aws::String statusString =
                    Aws::IAM::Model::StatusTypeMapper::GetNameForStatusType(
                            key.GetStatus());
            std::cout << std::left << std::setw(32) << key.GetUserName() <<
                      std::setw(30) << key.GetAccessKeyId() << std::setw(20) <<
                      statusString << std::setw(20) <<
                      key.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
 std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 ListAccessKeys。

CLI

Amazon CLI

列出 IAM 用户的访问密钥 ID

以下 `list-access-keys` 命令将列出名为 Bob 的 IAM 用户的访问密钥 ID。

```
aws iam list-access-keys \
    --user-name Bob
```

输出：

```
{
    "AccessKeyMetadata": [
        {
            "UserName": "Bob",
            "Status": "Active",
            "CreateDate": "2013-06-04T18:17:34Z",
            "AccessKeyId": "AKIAIOSFODNN7EXAMPLE"
        },
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CreateDate": "2013-06-06T20:42:26Z",
            "AccessKeyId": "AKIAI44QH8DHBEXAMPLE"
        }
    ]
}
```

无法列出 IAM 用户的秘密访问密钥。如果秘密访问密钥丢失，则必须使用 `create-access-keys` 命令创建新的访问密钥。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的访问密钥。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListAccessKeys。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListAccessKeys lists the access keys for the specified user.
func (wrapper UserWrapper) ListAccessKeys(ctx context.Context, userName string)
 ([]types.AccessKeyMetadata, error) {
 var keys []types.AccessKeyMetadata
 result, err := wrapper.IamClient.ListAccessKeys(ctx, &iam.ListAccessKeysInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list access keys for user %v. Here's why: %v\n", userName,
 err)
 } else {
```

```
  keys = result.AccessKeyMetadata
 }
 return keys, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 ListAccessKeys。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.AccessKeyMetadata;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccessKeysRequest;
import software.amazon.awssdk.services.iam.model.ListAccessKeysResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccessKeys {
    public static void main(String[] args) {
        final String usage = """

                Usage:
```

```
                    <userName>\s

                Where:
                    userName - The name of the user for which access keys are
retrieved.\s
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userName = args[0];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listKeys(iam, userName);
        System.out.println("Done");
        iam.close();
    }

    public static void listKeys(IamClient iam, String userName) {
        try {
            boolean done = false;
            String newMarker = null;

            while (!done) {
                ListAccessKeysResponse response;

                if (newMarker == null) {
                    ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                            .userName(userName)
                            .build();

                    response = iam.listAccessKeys(request);

                } else {
                    ListAccessKeysRequest request =
ListAccessKeysRequest.builder()
                            .userName(userName)
                            .marker(newMarker)
```

```
                            .build();

                    response = iam.listAccessKeys(request);
                }

                for (AccessKeyMetadata metadata : response.accessKeyMetadata()) {
                    System.out.format("Retrieved access key %s",
 metadata.accessKeyId());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 ListAccessKeys。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出访问密钥。

```
import { ListAccessKeysCommand, IAMClient } from "@aws-sdk/client-iam";
```

```
const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} userName
 */
export async function* listAccessKeys(userName) {
  const command = new ListAccessKeysCommand({
    MaxItems: 5,
    UserName: userName,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListAccessKeysCommandOutput |
 undefined}
   */
  let response = await client.send(command);

  while (response?.AccessKeyMetadata?.length) {
    for (const key of response.AccessKeyMetadata) {
      yield key;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccessKeysCommand({
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-listing。

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListAccessKeys。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 5,
  UserName: "IAM_USER_NAME",
};

iam.listAccessKeys(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-access-keys.html#iiam-examples-managing-access-keys-listing。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListAccessKeys。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun listKeys(userNameVal: String?) {
    val request =
        ListAccessKeysRequest {
            userName = userNameVal
        }
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccessKeys(request)
        response.accessKeyMetadata?.forEach { md ->
            println("Retrieved access key ${md.accessKeyId}")
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 ListAccessKeys。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令列出名为 **Bob** 的 IAM 用户的访问密钥。请注意，您无法列出 IAM 用户的秘密访问密钥。如果秘密访问密钥丢失，则必须使用 **New-IAMAccessKey** cmdlet 创建新的访问密钥。

```
Get-IAMAccessKey -UserName "Bob"
```

输出：

```
AccessKeyId               CreateDate                 Status
 UserName
-----------               ----------                 ------
 --------
AKIAIOSFODNN7EXAMPLE      12/3/2014 10:53:41 AM      Active             Bob
AKIAI44QH8DHBEXAMPLE      6/6/2013 8:42:26 PM        Inactive           Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListAccessKeys。

Tools for PowerShell V5

示例 1：此命令列出名为 **Bob** 的 IAM 用户的访问密钥。请注意，您无法列出 IAM 用户的秘密访问密钥。如果秘密访问密钥丢失，则必须使用 **New-IAMAccessKey** cmdlet 创建新的访问密钥。

```
Get-IAMAccessKey -UserName "Bob"
```

输出：

```
AccessKeyId               CreateDate                 Status
 UserName
-----------               ----------                 ------
 --------
AKIAIOSFODNN7EXAMPLE      12/3/2014 10:53:41 AM      Active             Bob
AKIAI44QH8DHBEXAMPLE      6/6/2013 8:42:26 PM        Inactive           Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListAccessKeys。

Python

适用于 Python 的 SDK (Boto3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListAccessKeys。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、停用和删除访问密钥。

```
# Manages access keys for IAM users
class AccessKeyManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'AccessKeyManager'
  end
```

```
 # Lists access keys for a user
 #
 # @param user_name [String] The name of the user.
 def list_access_keys(user_name)
   response = @iam_client.list_access_keys(user_name: user_name)
   if response.access_key_metadata.empty?
     @logger.info("No access keys found for user '#{user_name}'.")
   else
     response.access_key_metadata.map(&:access_key_id)
   end
 rescue Aws::IAM::Errors::NoSuchEntity
   @logger.error("Error listing access keys: cannot find user '#{user_name}'.")
   []
 rescue StandardError => e
   @logger.error("Error listing access keys: #{e.message}")
   []
 end

 # Creates an access key for a user
 #
 # @param user_name [String] The name of the user.
 # @return [Boolean]
 def create_access_key(user_name)
   response = @iam_client.create_access_key(user_name: user_name)
   access_key = response.access_key
   @logger.info("Access key created for user '#{user_name}':
#{access_key.access_key_id}")
   access_key
 rescue Aws::IAM::Errors::LimitExceeded
   @logger.error('Error creating access key: limit exceeded. Cannot create
more.')
   nil
 rescue StandardError => e
   @logger.error("Error creating access key: #{e.message}")
   nil
 end

 # Deactivates an access key
 #
 # @param user_name [String] The name of the user.
 # @param access_key_id [String] The ID for the access key.
 # @return [Boolean]
 def deactivate_access_key(user_name, access_key_id)
```

```
    @iam_client.update_access_key(
      user_name: user_name,
      access_key_id: access_key_id,
      status: 'Inactive'
    )
    true
  rescue StandardError => e
    @logger.error("Error deactivating access key: #{e.message}")
    false
  end

  # Deletes an access key
  #
  # @param user_name [String] The name of the user.
  # @param access_key_id [String] The ID for the access key.
  # @return [Boolean]
  def delete_access_key(user_name, access_key_id)
    @iam_client.delete_access_key(
      user_name: user_name,
      access_key_id: access_key_id
    )
    true
  rescue StandardError => e
    @logger.error("Error deleting access key: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 ListAccessKeys。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listaccesskeys(
          iv_username = iv_user_name ).
        MESSAGE 'Retrieved access key list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListAccessKeys。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListAccountAliases** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListAccountAliases。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理您的账户

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool
AwsDoc::IAM::listAccountAliases(const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListAccountAliasesRequest request;
```

```
    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListAccountAliases(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list account aliases: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        const auto &aliases = outcome.GetResult().GetAccountAliases();
        if (!header) {
            if (aliases.size() == 0) {
                std::cout << "Account has no aliases" << std::endl;
                break;
            }
            std::cout << std::left << std::setw(32) << "Alias" << std::endl;
            header = true;
        }

        for (const auto &alias: aliases) {
            std::cout << std::left << std::setw(32) << alias << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 ListAccountAliases。

CLI

Amazon CLI

列出账户别名

以下 `list-account-aliases` 命令将列出当前账户的别名。

```
aws iam list-account-aliases
```

输出：

```
{
    "AccountAliases": [
    "mycompany"
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的您的 Amazon 账户 ID 及其别名。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListAccountAliases。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListAccountAliasesResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListAccountAliases {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAliases(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAliases(IamClient iam) {
        try {
            ListAccountAliasesResponse response = iam.listAccountAliases();
            for (String alias : response.accountAliases()) {
                System.out.printf("Retrieved account alias %s", alias);
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 ListAccountAliases。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出账户别名。

```
import { ListAccountAliasesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listAccountAliases() {
  const command = new ListAccountAliasesCommand({ MaxItems: 5 });

  let response = await client.send(command);

  while (response.AccountAliases?.length) {
    for (const alias of response.AccountAliases) {
      yield alias;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAccountAliasesCommand({
          Marker: response.Marker,
          MaxItems: 5,
        }),
      );
    } else {
      break;
    }
  }
```

```
}
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-listing。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListAccountAliases。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listAccountAliases({ MaxItems: 10 }, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-account-aliases.html#iam-examples-account-aliases-listing。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListAccountAliases。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun listAliases() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listAccountAliases(ListAccountAliasesRequest {})
        response.accountAliases?.forEach { alias ->
            println("Retrieved account alias $alias")
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 ListAccountAliases。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回 Amazon Web Services 账户 的账户别名。

```
Get-IAMAccountAlias
```

输出：

```
ExampleCo
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListAccountAliases。

Tools for PowerShell V5

示例 1：此命令返回 Amazon Web Services 账户 的账户别名。

```
Get-IAMAccountAlias
```

输出：

```
ExampleCo
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListAccountAliases。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListAccountAliases。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、创建和删除账户别名。

```
class IAMAliasManager
  # Initializes the IAM client and logger
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client.
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists available AWS account aliases.
  def list_aliases
    response = @iam_client.list_account_aliases

    if response.account_aliases.count.positive?
      @logger.info('Account aliases are:')
      response.account_aliases.each { |account_alias| @logger.info("
 #{account_alias}") }
    else
      @logger.info('No account aliases found.')
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing account aliases: #{e.message}")
  end
```

```
  # Creates an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to create.
  # @return [Boolean] true if the account alias was created; otherwise, false.
  def create_account_alias(account_alias)
    @iam_client.create_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating account alias: #{e.message}")
    false
  end

  # Deletes an AWS account alias.
  #
  # @param account_alias [String] The name of the account alias to delete.
  # @return [Boolean] true if the account alias was deleted; otherwise, false.
  def delete_account_alias(account_alias)
    @iam_client.delete_account_alias(account_alias: account_alias)
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting account alias: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 ListAccountAliases。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listaccountaliases( ).
```

```
        MESSAGE 'Retrieved account alias list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing account aliases.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListAccountAliases。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListAttachedGroupPolicies** 与 CLI 配合使用

以下代码示例演示如何使用 ListAttachedGroupPolicies。

CLI

Amazon CLI

列出附加到指定组的所有托管策略

此示例将返回附加到 Amazon 账户中名为 Admins 的 IAM 组的托管策略名称和 ARN。

```
aws iam list-attached-group-policies \
    --group-name Admins
```

输出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
```

```
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListAttachedGroupPolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回附加到 Amazon 账户中名为 **Admins** 的 IAM 组的托管策略名称和 ARN。要查看组中嵌入的内联策略列表，请使用 **Get-IAMGroupPolicyList** 命令。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListAttachedGroupPolicies。

Tools for PowerShell V5

示例 1：此命令返回附加到 Amazon 账户中名为 **Admins** 的 IAM 组的托管策略名称和 ARN。要查看组中嵌入的内联策略列表，请使用 **Get-IAMGroupPolicyList** 命令。

```
Get-IAMAttachedGroupPolicyList -GroupName "Admins"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
arn:aws:iam::aws:policy/AdministratorAccess               AdministratorAccess
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListAttachedGroupPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListAttachedRolePolicies** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListAttachedRolePolicies。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List the IAM role policies that are attached to an IAM role.
    /// </summary>
    /// <param name="roleName">The IAM role to list IAM policies for.</param>
    /// <returns>A list of the IAM policies attached to the IAM role.</returns>
    public async Task<List<AttachedPolicyType>>
ListAttachedRolePoliciesAsync(string roleName)
    {
        var attachedPolicies = new List<AttachedPolicyType>();
        var attachedRolePoliciesPaginator =
_IAMService.Paginators.ListAttachedRolePolicies(new
ListAttachedRolePoliciesRequest { RoleName = roleName });

        await foreach (var response in attachedRolePoliciesPaginator.Responses)
        {
            attachedPolicies.AddRange(response.AttachedPolicies);
        }

        return attachedPolicies;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListAttachedRolePolicies。

CLI

Amazon CLI

列出附加到指定角色的所有托管策略

此命令将返回附加到 Amazon 账户中名为 SecurityAuditRole 的 IAM 角色的托管策略名称和 ARN。

```
aws iam list-attached-role-policies \
    --role-name SecurityAuditRole
```

输出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListAttachedRolePolicies。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
```

```
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListAttachedRolePolicies lists the policies that are attached to the specified
 role.
func (wrapper RoleWrapper) ListAttachedRolePolicies(ctx context.Context, roleName
 string) ([]types.AttachedPolicy, error) {
 var policies []types.AttachedPolicy
 result, err := wrapper.IamClient.ListAttachedRolePolicies(ctx,
 &iam.ListAttachedRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list attached policies for role %v. Here's why: %v\n",
 roleName, err)
 } else {
  policies = result.AttachedPolicies
 }
 return policies, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListAttachedRolePolicies。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出附加到角色的策略。

```
import {
  ListAttachedRolePoliciesCommand,
  IAMClient,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 * @param {string} roleName
 */
export async function* listAttachedRolePolicies(roleName) {
  const command = new ListAttachedRolePoliciesCommand({
    RoleName: roleName,
  });

  let response = await client.send(command);

  while (response.AttachedPolicies?.length) {
    for (const policy of response.AttachedPolicies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListAttachedRolePoliciesCommand({
          RoleName: roleName,
          Marker: response.Marker,
```

```
        }),
      );
    } else {
      break;
    }
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListAttachedRolePolicies。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listAttachedRolePolicies($roleName, $pathPrefix = "", $marker
 = "", $maxItems = 0)
    {
        $listAttachRolePoliciesArguments = ['RoleName' => $roleName];
        if ($pathPrefix) {
            $listAttachRolePoliciesArguments['PathPrefix'] = $pathPrefix;
        }
        if ($marker) {
            $listAttachRolePoliciesArguments['Marker'] = $marker;
        }
        if ($maxItems) {
            $listAttachRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->iamClient-
>listAttachedRolePolicies($listAttachRolePoliciesArguments);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListAttachedRolePolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回附加到 Amazon 账户中名为 **SecurityAuditRole** 的 IAM 角色的托管策略名称和 ARN。要查看嵌入在角色中的内联策略的列表，请使用 **Get-IAMRolePolicyList** 命令。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListAttachedRolePolicies。

Tools for PowerShell V5

示例 1：此命令返回附加到 Amazon 账户中名为 **SecurityAuditRole** 的 IAM 角色的托管策略名称和 ARN。要查看嵌入在角色中的内联策略的列表，请使用 **Get-IAMRolePolicyList** 命令。

```
Get-IAMAttachedRolePolicyList -RoleName "SecurityAuditRole"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/SecurityAudit                     SecurityAudit
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListAttachedRolePolicies。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_attached_policies(role_name):
    """
    Lists policies attached to a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.attached_policies.all():
            logger.info("Got policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't list attached policies for %s.", role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListAttachedRolePolicies。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
 #{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
 exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
```

```
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListAttachedRolePolicies。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_attached_role_policies(
    client: &iamClient,
    role_name: String,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListAttachedRolePoliciesOutput,
 SdkError<ListAttachedRolePoliciesError>> {
    let response = client
        .list_attached_role_policies()
        .role_name(role_name)
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListAttachedRolePolicies。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->listattachedrolepolicies(
      iv_rolename = iv_role_name ).
    MESSAGE 'Retrieved attached policy list for role.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListAttachedRolePolicies。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3
```

```
    /// Returns a list of AWS Identity and Access Management (IAM) policies
    /// that are attached to the role.
    ///
    /// - Parameter role: The IAM role to return the policy list for.
    ///
    /// - Returns: An array of `IAMClientTypes.AttachedPolicy` objects
    ///   describing each managed policy that's attached to the role.
    public func listAttachedRolePolicies(role: String) async throws ->
[IAMClientTypes.AttachedPolicy] {
        var policyList: [IAMClientTypes.AttachedPolicy] = []

        // Use "Paginated" to get all the attached role polices.
        // This lets the SDK handle the 'isTruncated' in
"ListAttachedRolePoliciesOutput".
        let input = ListAttachedRolePoliciesInput(
            roleName: role
        )
        let output = client.listAttachedRolePoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let attachedPolicies = page.attachedPolicies else {
                    print("Error: no attached policies returned.")
                    continue
                }
                for attachedPolicy in attachedPolicies {
                    policyList.append(attachedPolicy)
                }
            }
        } catch {
            print("ERROR: listAttachedRolePolicies:", dump(error))
            throw error
        }

        return policyList
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Rust API 参考中的 ListAttachedRolePolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListAttachedUserPolicies** 与 CLI 配合使用

以下代码示例演示如何使用 ListAttachedUserPolicies。

CLI

Amazon CLI

列出附加到指定用户的所有托管策略

此命令将返回附加到 Amazon 账户中名为 Bob 的 IAM 用户的托管策略名称和 ARN。

```
aws iam list-attached-user-policies \
    --user-name Bob
```

输出：

```
{
    "AttachedPolicies": [
        {
            "PolicyName": "AdministratorAccess",
            "PolicyArn": "arn:aws:iam::aws:policy/AdministratorAccess"
        },
        {
            "PolicyName": "SecurityAudit",
            "PolicyArn": "arn:aws:iam::aws:policy/SecurityAudit"
        }
    ],
    "IsTruncated": false
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListAttachedUserPolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此命令返回附加到 Amazon 账户中名为 **Bob** 的 IAM 用户的托管策略名称和 ARN。要查看嵌入在 IAM 用户中的内联策略的列表，请使用 **Get-IAMUserPolicyList** 命令。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/TesterPolicy                      TesterPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListAttachedUserPolicies。

Tools for PowerShell V5

示例 1：此命令返回附加到 Amazon 账户中名为 **Bob** 的 IAM 用户的托管策略名称和 ARN。要查看嵌入在 IAM 用户中的内联策略的列表，请使用 **Get-IAMUserPolicyList** 命令。

```
Get-IAMAttachedUserPolicyList -UserName "Bob"
```

输出：

```
PolicyArn                                                 PolicyName
---------                                                 ----------
arn:aws:iam::aws:policy/TesterPolicy                      TesterPolicy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListAttachedUserPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListEntitiesForPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 `ListEntitiesForPolicy`。

CLI

Amazon CLI

列出指定托管策略所附加到的所有用户、组和角色

此示例返回附加了策略 arn:aws:iam::123456789012:policy/TestPolicy 的 IAM 组、角色和用户的列表。

```
aws iam list-entities-for-policy \
    --policy-arn arn:aws:iam::123456789012:policy/TestPolicy
```

输出：

```
{
    "PolicyGroups": [
        {
            "GroupName": "Admins",
            "GroupId": "AGPACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyUsers": [
        {
            "UserName": "Alice",
            "UserId": "AIDACKCEVSQ6C2EXAMPLE"
        }
    ],
    "PolicyRoles": [
        {
            "RoleName": "DevRole",
            "RoleId": "AROADBQP57FF2AEXAMPLE"
        }
    ],
    "IsTruncated": false
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListEntitiesForPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回附加了策略 **arn:aws:iam::123456789012:policy/TestPolicy** 的 IAM 组、角色和用户的列表。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

输出：

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListEntitiesForPolicy。

Tools for PowerShell V5

示例 1：此示例返回附加了策略 **arn:aws:iam::123456789012:policy/TestPolicy** 的 IAM 组、角色和用户的列表。

```
Get-IAMEntitiesForPolicy -PolicyArn "arn:aws:iam::123456789012:policy/TestPolicy"
```

输出：

```
IsTruncated  : False
Marker       :
PolicyGroups : {}
PolicyRoles  : {testRole}
PolicyUsers  : {Bob, Theresa}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListEntitiesForPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListGroupPolicies** 与 CLI 配合使用

以下代码示例演示如何使用 `ListGroupPolicies`。

CLI

Amazon CLI

列出附加到指定组的所有内联策略

以下 `list-group-policies` 命令列出附加到当前文档中名为 `Admins` 的 IAM 组的内联策略名称。

```
aws iam list-group-policies \
    --group-name Admins
```

输出：

```
{
    "PolicyNames": [
        "AdminRoot",
        "ExamplePolicy"
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListGroupPolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回嵌入组 **Testers** 中的内联策略的列表。要获取附加到该组的托管策略，请使用命令 **Get-IAMAttachedGroupPolicyList**。

```
Get-IAMGroupPolicyList -GroupName Testers
```

输出：

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListGroupPolicies。

Tools for PowerShell V5

示例 1：返回嵌入组 **Testers** 中的内联策略的列表。要获取附加到该组的托管策略，请使用命令 **Get-IAMAttachedGroupPolicyList**。

```
Get-IAMGroupPolicyList -GroupName Testers
```

输出：

```
Deny-Assume-S3-Role-In-Production
PowerUserAccess-Testers
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListGroupPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListGroups** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListGroups。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List IAM groups.
    /// </summary>
    /// <returns>A list of IAM groups.</returns>
    public async Task<List<Group>> ListGroupsAsync()
    {
        var groupsPaginator = _IAMService.Paginators.ListGroups(new
ListGroupsRequest());
        var groups = new List<Group>();

        await foreach (var response in groupsPaginator.Responses)
        {
            groups.AddRange(response.Groups);
```

```
        }

        return groups;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListGroups。

CLI

Amazon CLI

列出当前账户的 IAM 组

以下 `list-groups` 命令将列出当前账户中的 IAM 组。

```
aws iam list-groups
```

输出：

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-06-04T20:27:27.972Z",
            "GroupId": "AIDACKCEVSQ6C2EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admins",
            "GroupName": "Admins"
        },
        {
            "Path": "/",
            "CreateDate": "2013-04-16T20:30:42Z",
            "GroupId": "AIDGPMS9RO4H3FEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/S3-Admins",
            "GroupName": "S3-Admins"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListGroups。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// GroupWrapper encapsulates AWS Identity and Access Management (IAM) group
 actions
// used in the examples.
// It contains an IAM service client that is used to perform group actions.
type GroupWrapper struct {
 IamClient *iam.Client
}



// ListGroups lists up to maxGroups number of groups.
func (wrapper GroupWrapper) ListGroups(ctx context.Context, maxGroups int32)
 ([]types.Group, error) {
 var groups []types.Group
 result, err := wrapper.IamClient.ListGroups(ctx, &iam.ListGroupsInput{
  MaxItems: aws.Int32(maxGroups),
 })
 if err != nil {
  log.Printf("Couldn't list groups. Here's why: %v\n", err)
 } else {
```

```
  groups = result.Groups
 }
 return groups, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListGroups。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出组。

```
import { ListGroupsCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 */
export async function* listGroups() {
  const command = new ListGroupsCommand({
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.Groups?.length) {
    for (const group of response.Groups) {
      yield group;
    }
```

```
    if (response.IsTruncated) {
      response = await client.send(
        new ListGroupsCommand({
          Marker: response.Marker,
          MaxItems: 10,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListGroups。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listGroups($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listGroupsArguments = [];
        if ($pathPrefix) {
            $listGroupsArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listGroupsArguments["Marker"] = $marker;
        }
        if ($maxItems) {
```

```
            $listGroupsArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listGroups($listGroupsArguments);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListGroups。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 Amazon Web Services 账户 中定义的所有 IAM 组的集合。

```
Get-IAMGroupList
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListGroups。

Tools for PowerShell V5

示例 1：此示例返回当前 Amazon Web Services 账户 中定义的所有 IAM 组的集合。

```
Get-IAMGroupList
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE2
GroupName  : Developers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE3
GroupName  : Testers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListGroups。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_groups(count):
    """
    Lists the specified number of groups for the account.

    :param count: The number of groups to list.
```

```
    """
    try:
        for group in iam.groups.limit(count):
            logger.info("Group: %s", group.name)
    except ClientError:
        logger.exception("Couldn't list groups for the account.")
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListGroups。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# A class to manage IAM operations via the AWS SDK client
class IamGroupManager
  # Initializes the IamGroupManager class
  # @param iam_client [Aws::IAM::Client] An instance of the IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of groups for the account.
  # @param count [Integer] The maximum number of groups to list.
  # @return [Aws::IAM::Client::Response]
  def list_groups(count)
    response = @iam_client.list_groups(max_items: count)
    response.groups.each do |group|
      @logger.info("\t#{group.group_name}")
    end
```

```
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list groups for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListGroups。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_groups(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListGroupsOutput, SdkError<ListGroupsError>> {
    let response = client
        .list_groups()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListGroups。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listgroups( ).
        MESSAGE 'Retrieved group list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing groups.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListGroups。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func listGroups() async throws -> [String] {
        var groupList: [String] = []

        // Use "Paginated" to get all the groups.
```

```
        // This lets the SDK handle the 'isTruncated' property in
"ListGroupsOutput".
        let input = ListGroupsInput()

        let pages = client.listGroupsPaginated(input: input)
        do {
            for try await page in pages {
                guard let groups = page.groups else {
                    print("Error: no groups returned.")
                    continue
                }

                for group in groups {
                    if let name = group.groupName {
                        groupList.append(name)
                    }
                }
            }
        } catch {
            print("ERROR: listGroups:", dump(error))
            throw error
        }
        return groupList
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 ListGroups。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListGroupsForUser** 与 CLI 配合使用

以下代码示例演示如何使用 `ListGroupsForUser`。

CLI

Amazon CLI

列出 IAM 用户所属的组

以下 `list-groups-for-user` 命令显示名为 Bob 的 IAM 用户所属的组。

```
aws iam list-groups-for-user \
    --user-name Bob
```

输出：

```
{
    "Groups": [
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:18:08Z",
            "GroupId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/Admin",
            "GroupName": "Admin"
        },
        {
            "Path": "/",
            "CreateDate": "2013-05-06T01:37:28Z",
            "GroupId": "AKIAI44QH8DHBEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:group/s3-Users",
            "GroupName": "s3-Users"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListGroupsForUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回 IAM 用户 **David** 所属的 IAM 组列表。

```
Get-IAMGroupForUser -UserName David
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
```

```
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListGroupsForUser。

Tools for PowerShell V5

示例 1：此示例返回 IAM 用户 **David** 所属的 IAM 组列表。

```
Get-IAMGroupForUser -UserName David
```

输出：

```
Arn        : arn:aws:iam::123456789012:group/Administrators
CreateDate : 10/20/2014 10:06:24 AM
GroupId    : 6WCH4TRY3KIHIEXAMPLE1
GroupName  : Administrators
Path       : /

Arn        : arn:aws:iam::123456789012:group/Testers
CreateDate : 12/10/2014 3:39:11 PM
GroupId    : RHNZZGQJ7QHMAEXAMPLE2
GroupName  : Testers
Path       : /

Arn        : arn:aws:iam::123456789012:group/Developers
CreateDate : 12/10/2014 3:38:55 PM
GroupId    : ZU2EOWMK6WBZOEXAMPLE3
GroupName  : Developers
Path       : /
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListGroupsForUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListInstanceProfiles** 与 CLI 配合使用

以下代码示例演示如何使用 `ListInstanceProfiles`。

CLI

Amazon CLI

列出账户的实例配置文件

以下 `list-instance-profiles` 命令列出与当前账户关联的实例配置文件。

```
aws iam list-instance-profiles
```

输出：

```
{
    "InstanceProfiles": [
        {
            "Path": "/",
            "InstanceProfileName": "example-dev-role",
            "InstanceProfileId": "AIPAIXEU4NUHUPEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-dev-role",
            "CreateDate": "2023-09-21T18:17:41+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-dev-role",
                    "RoleId": "AROAJ52OTH4H7LEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-dev-role",
                    "CreateDate": "2023-09-21T18:17:40+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
```

```
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        },
        {
            "Path": "/",
            "InstanceProfileName": "example-s3-role",
            "InstanceProfileId": "AIPAJVJVNRIQFREXAMPLE",
            "Arn": "arn:aws:iam::123456789012:instance-profile/example-s3-role",
            "CreateDate": "2023-09-21T18:18:50+00:00",
            "Roles": [
                {
                    "Path": "/",
                    "RoleName": "example-s3-role",
                    "RoleId": "AROAINUBC5O7XLEXAMPLE",
                    "Arn": "arn:aws:iam::123456789012:role/example-s3-role",
                    "CreateDate": "2023-09-21T18:18:49+00:00",
                    "AssumeRolePolicyDocument": {
                        "Version":"2012-10-17",
                        "Statement": [
                            {
                                "Effect": "Allow",
                                "Principal": {
                                    "Service": "ec2.amazonaws.com"
                                },
                                "Action": "sts:AssumeRole"
                            }
                        ]
                    }
                }
            ]
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListInstanceProfiles。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 Amazon Web Services 账户 中定义的实例配置文件的集合。

```
Get-IAMInstanceProfileList
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListInstanceProfiles。

Tools for PowerShell V5

示例 1：此示例返回当前 Amazon Web Services 账户 中定义的实例配置文件的集合。

```
Get-IAMInstanceProfileList
```

输出：

```
Arn                 : arn:aws:iam::123456789012:instance-profile/ec2instancerole
CreateDate          : 2/17/2015 2:49:04 PM
InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
InstanceProfileName : ec2instancerole
Path                : /
Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListInstanceProfiles。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListInstanceProfilesForRole** 与 CLI 配合使用

以下代码示例演示如何使用 ListInstanceProfilesForRole。

CLI

Amazon CLI

列出 IAM 角色的实例配置文件

以下 list-instance-profiles-for-role 命令列出了中与角色 Test-Role 关联的实例配置文件。

```
aws iam list-instance-profiles-for-role \
    --role-name Test-Role
```

输出：

```
{
    "InstanceProfiles": [
        {
            "InstanceProfileId": "AIDGPMS9RO4H3FEXAMPLE",
            "Roles": [
                {
                    "AssumeRolePolicyDocument": "<URL-encoded-JSON>",
                    "RoleId": "AIDACKCEVSQ6C2EXAMPLE",
                    "CreateDate": "2013-06-07T20:42:15Z",
                    "RoleName": "Test-Role",
                    "Path": "/",
                    "Arn": "arn:aws:iam::123456789012:role/Test-Role"
                }
            ],
            "CreateDate": "2013-06-07T21:05:24Z",
            "InstanceProfileName": "ExampleInstanceProfile",
            "Path": "/",
            "Arn": "arn:aws:iam::123456789012:instance-profile/
ExampleInstanceProfile"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListInstanceProfilesForRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回与角色 **ec2instancerole** 关联的实例配置文件的详细信息。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

输出：

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListInstanceProfilesForRole。

Tools for PowerShell V5

示例 1：此示例返回与角色 **ec2instancerole** 关联的实例配置文件的详细信息。

```
Get-IAMInstanceProfileForRole -RoleName ec2instancerole
```

输出：

```
      Arn                 : arn:aws:iam::123456789012:instance-profile/
ec2instancerole
      CreateDate          : 2/17/2015 2:49:04 PM
      InstanceProfileId   : HH36PTZQJUR32EXAMPLE1
      InstanceProfileName : ec2instancerole
      Path                : /
      Roles               : {ec2instancerole}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListInstanceProfilesForRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListMfaDevices** 与 CLI 配合使用

以下代码示例演示如何使用 ListMfaDevices。

CLI

Amazon CLI

列出指定用户的所有 MFA 设备

此示例返回有关分配给 IAM 用户 Bob 的 MFA 设备的详细信息。

```
aws iam list-mfa-devices \
    --user-name Bob
```

输出：

```
{
    "MFADevices": [
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Bob",
            "EnableDate": "2019-10-28T20:37:09+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "GAKT12345678",
            "EnableDate": "2023-02-18T21:44:42+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey1-7XNL7NFNLZ123456789EXAMPLE",
            "EnableDate": "2023-09-19T02:25:35+00:00"
        },
        {
            "UserName": "Bob",
            "SerialNumber": "arn:aws:iam::123456789012:u2f/user/Bob/
fidosecuritykey2-VDRQTDBBN5123456789EXAMPLE",
            "EnableDate": "2023-09-19T01:49:18+00:00"
        }
```

```
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 中使用多重身份验证（MFA）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListMfaDevices。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回有关分配给 IAM 用户 **David** 的 MFA 设备的详细信息。在此示例中，您可以看出它是虚拟设备，因为 **SerialNumber** 是 ARN，而不是物理设备的实际序列号。

```
Get-IAMMFADevice -UserName David
```

输出：

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListMfaDevices。

Tools for PowerShell V5

示例 1：此示例返回有关分配给 IAM 用户 **David** 的 MFA 设备的详细信息。在此示例中，您可以看出它是虚拟设备，因为 **SerialNumber** 是 ARN，而不是物理设备的实际序列号。

```
Get-IAMMFADevice -UserName David
```

输出：

```
EnableDate                SerialNumber                            UserName
----------                ------------                            --------
4/8/2015 9:41:10 AM       arn:aws:iam::123456789012:mfa/David     David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListMfaDevices。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListOpenIdConnectProviders** 与 CLI 配合使用

以下代码示例演示如何使用 ListOpenIdConnectProviders。

CLI

Amazon CLI

列出 Amazon 账户中 OpenID Connect 提供者的相关信息

此示例返回当前 Amazon 账户中定义的所有 OpenID Connect 提供者的 ARN 列表。

```
aws iam list-open-id-connect-providers
```

输出：

```
{
    "OpenIDConnectProviderList": [
        {
            "Arn": "arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListOpenIdConnectProviders。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回当前 Amazon Web Services 账户 账户中定义的所有 OpenID Connect 提供者的 ARN 列表。

```
Get-IAMOpenIDConnectProviderList
```

输出：

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListOpenIdConnectProviders。

Tools for PowerShell V5

示例 1：此示例返回当前 Amazon Web Services 账户 账户中定义的所有 OpenID Connect 提供者的 ARN 列表。

```
Get-IAMOpenIDConnectProviderList
```

输出：

```
Arn
---
arn:aws:iam::123456789012:oidc-provider/server.example.com
arn:aws:iam::123456789012:oidc-provider/another.provider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListOpenIdConnectProviders。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListPolicies** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListPolicies。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理策略

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List IAM policies.
    /// </summary>
    /// <returns>A list of the IAM policies.</returns>
    public async Task<List<ManagedPolicy>> ListPoliciesAsync()
    {
        var listPoliciesPaginator = _IAMService.Paginators.ListPolicies(new
ListPoliciesRequest());
        var policies = new List<ManagedPolicy>();

        await foreach (var response in listPoliciesPaginator.Responses)
        {
            policies.AddRange(response.Policies);
        }

        return policies;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 ListPolicies。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::listPolicies(const Aws::Client::ClientConfiguration
 &clientConfig) {
    const Aws::String DATE_FORMAT("%Y-%m-%d");
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListPoliciesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListPolicies(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam policies: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(64) << "Description" << std::setw(12) <<
                      "CreateDate" << std::endl;
            header = true;
        }

        const auto &policies = outcome.GetResult().GetPolicies();
        for (const auto &policy: policies) {
            std::cout << std::left << std::setw(55) <<
                      policy.GetPolicyName() << std::setw(30) <<
                      policy.GetPolicyId() << std::setw(80) << policy.GetArn() <<
                      std::setw(64) << policy.GetDescription() << std::setw(12)
 <<
                      policy.GetCreateDate().ToGmtString(DATE_FORMAT.c_str()) <<
```

```
                        std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 ListPolicies。

CLI

Amazon CLI

列出您的 Amazon 账户可用的托管策略

此示例将返回当前 Amazon 账户中可用的前两个托管策略的集合。

```
aws iam list-policies \
    --max-items 3
```

输出：

```
{
    "Policies": [
        {
            "PolicyName": "AWSCloudTrailAccessPolicy",
            "PolicyId": "ANPAXQE2B5PJ7YEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:policy/AWSCloudTrailAccessPolicy",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 0,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
```

```
            "CreateDate": "2019-09-04T17:43:42+00:00",
            "UpdateDate": "2019-09-04T17:43:42+00:00"
        },
        {
            "PolicyName": "AdministratorAccess",
            "PolicyId": "ANPAIWMBCKSKIEE64ZLYK",
            "Arn": "arn:aws:iam::aws:policy/AdministratorAccess",
            "Path": "/",
            "DefaultVersionId": "v1",
            "AttachmentCount": 6,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:46+00:00",
            "UpdateDate": "2015-02-06T18:39:46+00:00"
        },
        {
            "PolicyName": "PowerUserAccess",
            "PolicyId": "ANPAJYRXTHIB4FOVS3ZXS",
            "Arn": "arn:aws:iam::aws:policy/PowerUserAccess",
            "Path": "/",
            "DefaultVersionId": "v5",
            "AttachmentCount": 1,
            "PermissionsBoundaryUsageCount": 0,
            "IsAttachable": true,
            "CreateDate": "2015-02-06T18:39:47+00:00",
            "UpdateDate": "2023-07-06T22:04:00+00:00"
        }
    ],
    "NextToken": "EXAMPLErZXIiOiBudWxsLCAiYm90b190cnVuY2F0ZV9hbW91bnQiOiA4fQ=="
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListPolicies。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// PolicyWrapper encapsulates AWS Identity and Access Management (IAM) policy
 actions
// used in the examples.
// It contains an IAM service client that is used to perform policy actions.
type PolicyWrapper struct {
 IamClient *iam.Client
}



// ListPolicies gets up to maxPolicies policies.
func (wrapper PolicyWrapper) ListPolicies(ctx context.Context, maxPolicies int32)
 ([]types.Policy, error) {
 var policies []types.Policy
 result, err := wrapper.IamClient.ListPolicies(ctx, &iam.ListPoliciesInput{
  MaxItems: aws.Int32(maxPolicies),
 })
 if err != nil {
  log.Printf("Couldn't list policies. Here's why: %v\n", err)
 } else {
  policies = result.Policies
```

```
 }
 return policies, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 ListPolicies。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出策略。

```
import { ListPoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listPolicies() {
  const command = new ListPoliciesCommand({
    MaxItems: 10,
    OnlyAttached: false,
    // List only the customer managed policies in your Amazon Web Services
 account.
    Scope: "Local",
  });

  let response = await client.send(command);
```

```
  while (response.Policies?.length) {
    for (const policy of response.Policies) {
      yield policy;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListPoliciesCommand({
          Marker: response.Marker,
          MaxItems: 10,
          OnlyAttached: false,
          Scope: "Local",
        }),
      );
    } else {
      break;
    }
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListPolicies。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listPolicies($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listPoliciesArguments = [];
```

```
        if ($pathPrefix) {
            $listPoliciesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listPoliciesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listPoliciesArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listPolicies($listPoliciesArguments);
    }
```

- 有关 API 详细信息，请参阅《适用于 PHP 的 Amazon SDK API Reference》中的 ListPolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将返回当前 Amazon 账户中可用的前三个托管策略的集合。由于 **-scope** 未指定，因此其默认为 **all**，并包含 Amazon 托管策略和客户管理型策略。

```
Get-IAMPolicyList -MaxItem 3
```

输出：

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
```

```
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

示例 2：此示例将返回当前 Amazon 账户中可用的前两个客户管理型策略的集合。其使用 **-Scope local** 将输出限制为仅限客户管理型策略。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

输出：

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
```

```
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListPolicies。

Tools for PowerShell V5

示例 1：此示例将返回当前 Amazon 账户中可用的前三个托管策略的集合。由于 **-scope** 未指定，因此其默认为 **all**，并包含 Amazon 托管策略和客户管理型策略。

```
Get-IAMPolicyList -MaxItem 3
```

输出：

```
Arn              : arn:aws:iam::aws:policy/AWSDirectConnectReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:08 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : Z27SI6FQMGNQ2EXAMPLE1
PolicyName       : AWSDirectConnectReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:08 AM

Arn              : arn:aws:iam::aws:policy/AmazonGlacierReadOnlyAccess
AttachmentCount  : 0
CreateDate       : 2/6/2015 10:40:27 AM
DefaultVersionId : v1
Description      :
IsAttachable     : True
Path             : /
PolicyId         : NJKMU274MET4EEXAMPLE2
PolicyName       : AmazonGlacierReadOnlyAccess
UpdateDate       : 2/6/2015 10:40:27 AM

Arn              : arn:aws:iam::aws:policy/AWSMarketplaceFullAccess
AttachmentCount  : 0
CreateDate       : 2/11/2015 9:21:45 AM
DefaultVersionId : v1
```

```
Description      :
IsAttachable     : True
Path             : /
PolicyId         : 5ULJSO2FYVPYGEXAMPLE3
PolicyName       : AWSMarketplaceFullAccess
UpdateDate       : 2/11/2015 9:21:45 AM
```

示例 2：此示例将返回当前 Amazon 账户中可用的前两个客户管理型策略的集合。其使用 **-Scope local** 将输出限制为仅限客户管理型策略。

```
Get-IAMPolicyList -Scope local -MaxItem 2
```

输出：

```
Arn              : arn:aws:iam::123456789012:policy/MyLocalPolicy
AttachmentCount  : 0
CreateDate       : 2/12/2015 9:39:09 AM
DefaultVersionId : v2
Description      :
IsAttachable     : True
Path             : /
PolicyId         : SQVCBLC4VAOUCEXAMPLE4
PolicyName       : MyLocalPolicy
UpdateDate       : 2/12/2015 9:39:53 AM

Arn              : arn:aws:iam::123456789012:policy/policyforec2instancerole
AttachmentCount  : 1
CreateDate       : 2/17/2015 2:51:38 PM
DefaultVersionId : v11
Description      :
IsAttachable     : True
Path             : /
PolicyId         : X5JPBLJH2Z2SOEXAMPLE5
PolicyName       : policyforec2instancerole
UpdateDate       : 2/18/2015 8:52:31 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListPolicies。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 ListPolicies。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例模块会列出、创建、附加和分离角色策略。

```
# Manages policies in AWS Identity and Access Management (IAM)
class RolePolicyManager
  # Initialize with an AWS IAM client
  #
  # @param iam_client [Aws::IAM::Client] An initialized IAM client
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'PolicyManager'
  end

  # Creates a policy
  #
  # @param policy_name [String] The name of the policy
  # @param policy_document [Hash] The policy document
  # @return [String] The policy ARN if successful, otherwise nil
  def create_policy(policy_name, policy_document)
    response = @iam_client.create_policy(
      policy_name: policy_name,
      policy_document: policy_document.to_json
    )
    response.policy.arn
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error creating policy: #{e.message}")
    nil
  end

  # Fetches an IAM policy by its ARN
  # @param policy_arn [String] the ARN of the IAM policy to retrieve
  # @return [Aws::IAM::Types::GetPolicyResponse] the policy object if found
  def get_policy(policy_arn)
```

```
    response = @iam_client.get_policy(policy_arn: policy_arn)
    policy = response.policy
    @logger.info("Got policy '#{policy.policy_name}'. Its ID is:
#{policy.policy_id}.")
    policy
  rescue Aws::IAM::Errors::NoSuchEntity
    @logger.error("Couldn't get policy '#{policy_arn}'. The policy does not
exist.")
    raise
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't get policy '#{policy_arn}'. Here's why: #{e.code}:
#{e.message}")
    raise
  end

  # Attaches a policy to a role
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def attach_policy_to_role(role_name, policy_arn)
    @iam_client.attach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error attaching policy to role: #{e.message}")
    false
  end

  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end

  # Detaches a policy from a role
```

```
  #
  # @param role_name [String] The name of the role
  # @param policy_arn [String] The policy ARN
  # @return [Boolean] true if successful, false otherwise
  def detach_policy_from_role(role_name, policy_arn)
    @iam_client.detach_role_policy(
      role_name: role_name,
      policy_arn: policy_arn
    )
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error detaching policy from role: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 ListPolicies。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_policies(
    client: iamClient,
    path_prefix: String,
) -> Result<Vec<String>, SdkError<ListPoliciesError>> {
    let list_policies = client
        .list_policies()
        .path_prefix(path_prefix)
        .scope(PolicyScopeType::Local)
        .into_paginator()
        .items()
        .send()
```

```
        .try_collect()
        .await?;

    let policy_names = list_policies
        .into_iter()
        .map(|p| {
            let name = p
                .policy_name
                .unwrap_or_else(|| "Missing Policy Name".to_string());
            println!("{}", name);
            name
        })
        .collect();

    Ok(policy_names)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListPolicies。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listpolicies( iv_scope = iv_scope ).
        MESSAGE 'Retrieved policy list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing policies.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListPolicies。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func listPolicies() async throws -> [MyPolicyRecord] {
        var policyList: [MyPolicyRecord] = []

        // Use "Paginated" to get all the policies.
        // This lets the SDK handle the 'isTruncated' in "ListPoliciesOutput".
        let input = ListPoliciesInput()
        let output = client.listPoliciesPaginated(input: input)

        do {
            for try await page in output {
                guard let policies = page.policies else {
                    print("Error: no policies returned.")
                    continue
                }

                for policy in policies {
                    guard let name = policy.policyName,
                          let id = policy.policyId,
                          let arn = policy.arn
                    else {
                        throw ServiceHandlerError.noSuchPolicy
                    }
                    policyList.append(MyPolicyRecord(name: name, id: id, arn:
 arn))
                }
            }
        } catch {
            print("ERROR: listPolicies:", dump(error))
```

```
            throw error
        }

        return policyList
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 ListPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListPolicyVersions** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListPolicyVersions。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 管理策略
- 回滚策略版本

CLI

Amazon CLI

列出有关指定托管策略版本的信息

此示例返回 ARN 为 arn:aws:iam::123456789012:policy/MySamplePolicy 的策略的可用版本列表。

```
aws iam list-policy-versions \
    --policy-arn arn:aws:iam::123456789012:policy/MySamplePolicy
```

输出：

```
{
    "IsTruncated": false,
    "Versions": [
```

```
        {
        "VersionId": "v2",
        "IsDefaultVersion": true,
        "CreateDate": "2015-06-02T23:19:44Z"
        },
        {
        "VersionId": "v1",
        "IsDefaultVersion": false,
        "CreateDate": "2015-06-02T22:30:47Z"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListPolicyVersions。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略的可用版本列表。要获取特定版本的策略文档，请使用 **Get-IAMPolicyVersion** 命令并指定所需版本的 **VersionId**。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

输出：

```
CreateDate                  Document                IsDefaultVersion
       VersionId
----------                  --------                ----------------
       ---------
2/12/2015 9:39:53 AM                                True
       v2
2/12/2015 9:39:09 AM                                False
       v1
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListPolicyVersions。

Tools for PowerShell V5

示例 1：此示例返回 ARN 为 **arn:aws:iam::123456789012:policy/MyManagedPolicy** 的策略的可用版本列表。要获取特定版本的策略文档，请使用 **Get-IAMPolicyVersion** 命令并指定所需版本的 **VersionId**。

```
Get-IAMPolicyVersionList -PolicyArn arn:aws:iam::123456789012:policy/
MyManagedPolicy
```

输出：

```
CreateDate                    Document                IsDefaultVersion
       VersionId
----------                    --------                ----------------
       ---------
2/12/2015 9:39:53 AM                                  True
       v2
2/12/2015 9:39:09 AM                                  False
       v1
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListPolicyVersions。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listpolicyversions(
          iv_policyarn = iv_policy_arn ).
        MESSAGE 'Retrieved policy versions list.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamservicefailureex.
```

```
        MESSAGE 'Service failure when listing policy versions.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListPolicyVersions。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListRolePolicies** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListRolePolicies。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- IoT Device Defender 入门

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List IAM role policies.
    /// </summary>
    /// <param name="roleName">The IAM role for which to list IAM policies.</
param>
    /// <returns>A list of IAM policy names.</returns>
    public async Task<List<string>> ListRolePoliciesAsync(string roleName)
    {
        var listRolePoliciesPaginator =
 _IAMService.Paginators.ListRolePolicies(new ListRolePoliciesRequest { RoleName =
 roleName });
```

```
        var policyNames = new List<string>();

        await foreach (var response in listRolePoliciesPaginator.Responses)
        {
            policyNames.AddRange(response.PolicyNames);
        }

        return policyNames;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListRolePolicies。

CLI

Amazon CLI

列出附加到 IAM 角色的策略

以下 `list-role-policies` 命令将列出指定 IAM 角色的权限策略名称。

```
aws iam list-role-policies \
    --role-name Test-Role
```

输出：

```
{
    "PolicyNames": [
        "ExamplePolicy"
    ]
}
```

要查看附加到角色的信任策略，请使用 `get-role` 命令。要查看权限策略的详细信息，请使用 `get-role-policy` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListRolePolicies。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRolePolicies lists the inline policies for a role.
func (wrapper RoleWrapper) ListRolePolicies(ctx context.Context, roleName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListRolePolicies(ctx,
 &iam.ListRolePoliciesInput{
  RoleName: aws.String(roleName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for role %v. Here's why: %v\n", roleName,
 err)
 } else {
```

```
  policies = result.PolicyNames
 }
 return policies, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListRolePolicies。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出策略。

```
import { ListRolePoliciesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 * @param {string} roleName
 */
export async function* listRolePolicies(roleName) {
  const command = new ListRolePoliciesCommand({
    RoleName: roleName,
    MaxItems: 10,
  });

  let response = await client.send(command);

  while (response.PolicyNames?.length) {
```

```
    for (const policyName of response.PolicyNames) {
      yield policyName;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolePoliciesCommand({
          RoleName: roleName,
          MaxItems: 10,
          Marker: response.Marker,
        }),
      );
    } else {
      break;
    }
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListRolePolicies。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listRolePolicies($roleName, $marker = "", $maxItems = 0)
    {
        $listRolePoliciesArguments = ['RoleName' => $roleName];
        if ($marker) {
            $listRolePoliciesArguments['Marker'] = $marker;
        }
```

```
        if ($maxItems) {
            $listRolePoliciesArguments['MaxItems'] = $maxItems;
        }
        return $this->customWaiter(function () use ($listRolePoliciesArguments) {
            return $this->iamClient-
>listRolePolicies($listRolePoliciesArguments);
        });
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListRolePolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例返回 IAM 角色 **lamda_exec_role** 中嵌入的内联策略名称列表。要查看内联策略的详细信息，请使用 **Get-IAMRolePolicy** 命令。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

输出：

```
oneClick_lambda_exec_role_policy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListRolePolicies。

Tools for PowerShell V5

示例 1：此示例返回 IAM 角色 **lamda_exec_role** 中嵌入的内联策略名称列表。要查看内联策略的详细信息，请使用 **Get-IAMRolePolicy** 命令。

```
Get-IAMRolePolicyList -RoleName lambda_exec_role
```

输出：

```
oneClick_lambda_exec_role_policy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListRolePolicies。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_policies(role_name):
    """
    Lists inline policies for a role.

    :param role_name: The name of the role to query.
    """
    try:
        role = iam.Role(role_name)
        for policy in role.policies.all():
            logger.info("Got inline policy %s.", policy.name)
    except ClientError:
        logger.exception("Couldn't list inline policies for %s.", role_name)
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListRolePolicies。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
  # Lists policy ARNs attached to a role
  #
  # @param role_name [String] The name of the role
  # @return [Array<String>] List of policy ARNs
  def list_attached_policy_arns(role_name)
    response = @iam_client.list_attached_role_policies(role_name: role_name)
    response.attached_policies.map(&:policy_arn)
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing policies attached to role: #{e.message}")
    []
  end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListRolePolicies。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_role_policies(
    client: &iamClient,
    role_name: &str,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolePoliciesOutput, SdkError<ListRolePoliciesError>> {
    let response = client
        .list_role_policies()
        .role_name(role_name)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;

    Ok(response)
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListRolePolicies。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->listrolepolicies(
      iv_rolename = iv_role_name ).
    MESSAGE 'Retrieved inline policy list for role.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Role does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListRolePolicies。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
```

```
import AWSS3


    public func listRolePolicies(role: String) async throws -> [String] {
        var policyList: [String] = []

        // Use "Paginated" to get all the role policies.
        // This lets the SDK handle the 'isTruncated' in
 "ListRolePoliciesOutput".
        let input = ListRolePoliciesInput(
            roleName: role
        )
        let pages = client.listRolePoliciesPaginated(input: input)

        do {
            for try await page in pages {
                guard let policies = page.policyNames else {
                    print("Error: no role policies returned.")
                    continue
                }

                for policy in policies {
                    policyList.append(policy)
                }
            }
        } catch {
            print("ERROR: listRolePolicies:", dump(error))
            throw error
        }
        return policyList
    }
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 ListRolePolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListRoleTags** 与 CLI 配合使用

以下代码示例演示如何使用 ListRoleTags。

CLI

Amazon CLI

列出附加到角色的标签

以下 `list-role-tags` 命令检索与指定角色关联的标签列表。

```
aws iam list-role-tags \
    --role-name production-role
```

输出：

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListRoleTags。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例获取与角色关联的标签。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListRoleTags。

Tools for PowerShell V5

示例 1：此示例获取与角色关联的标签。

```
Get-IAMRoleTagList -RoleName MyRoleName
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListRoleTags。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListRoles** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListRoles。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List IAM roles.
    /// </summary>
    /// <returns>A list of IAM roles.</returns>
    public async Task<List<Role>> ListRolesAsync()
    {
        var listRolesPaginator = _IAMService.Paginators.ListRoles(new
ListRolesRequest());
        var roles = new List<Role>();

        await foreach (var response in listRolesPaginator.Responses)
        {
            roles.AddRange(response.Roles);
        }
```

```
        return roles;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListRoles。

CLI

Amazon CLI

列出当前账户的 IAM 角色

以下 `list-roles` 命令将列出当前账户中的 IAM 角色。

```
aws iam list-roles
```

输出：

```
{
    "Roles": [
        {
            "Path": "/",
            "RoleName": "ExampleRole",
            "RoleId": "AROAJ52OTH4H7LEXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/ExampleRole",
            "CreateDate": "2017-09-12T19:23:36+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        },
        {
```

```
            "Path": "/example_path/",
            "RoleName": "ExampleRoleWithPath",
            "RoleId": "AROAI4QRP7UFT7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:role/example_path/
ExampleRoleWithPath",
            "CreateDate": "2023-09-21T20:29:38+00:00",
            "AssumeRolePolicyDocument": {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Sid": "",
                        "Effect": "Allow",
                        "Principal": {
                            "Service": "ec2.amazonaws.com"
                        },
                        "Action": "sts:AssumeRole"
                    }
                ]
            },
            "MaxSessionDuration": 3600
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListRoles。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// RoleWrapper encapsulates AWS Identity and Access Management (IAM) role actions
// used in the examples.
// It contains an IAM service client that is used to perform role actions.
type RoleWrapper struct {
 IamClient *iam.Client
}



// ListRoles gets up to maxRoles roles.
func (wrapper RoleWrapper) ListRoles(ctx context.Context, maxRoles int32)
 ([]types.Role, error) {
 var roles []types.Role
 result, err := wrapper.IamClient.ListRoles(ctx,
  &iam.ListRolesInput{MaxItems: aws.Int32(maxRoles)},
 )
 if err != nil {
  log.Printf("Couldn't list roles. Here's why: %v\n", err)
 } else {
  roles = result.Roles
 }
 return roles, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListRoles。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出角色。

```
import { ListRolesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listRoles() {
  const command = new ListRolesCommand({
    MaxItems: 10,
  });

  /**
   * @type {import("@aws-sdk/client-iam").ListRolesCommandOutput | undefined}
   */
  let response = await client.send(command);

  while (response?.Roles?.length) {
    for (const role of response.Roles) {
      yield role;
    }

    if (response.IsTruncated) {
      response = await client.send(
        new ListRolesCommand({
          Marker: response.Marker,
        }),
```

```
      );
    } else {
      break;
    }
  }
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListRoles。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    /**
     * @param string $pathPrefix
     * @param string $marker
     * @param int $maxItems
     * @return Result
     * $roles = $service->listRoles();
     */
    public function listRoles($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listRolesArguments = [];
        if ($pathPrefix) {
            $listRolesArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listRolesArguments["Marker"] = $marker;
        }
        if ($maxItems) {
```

```
            $listRolesArguments["MaxItems"] = $maxItems;
        }
        return $this->iamClient->listRoles($listRolesArguments);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListRoles。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索 Amazon Web Services 账户 中的所有 IAM 角色列表。

```
Get-IAMRoleList
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListRoles。

Tools for PowerShell V5

示例 1：此示例检索 Amazon Web Services 账户 中的所有 IAM 角色列表。

```
Get-IAMRoleList
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListRoles。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_roles(count):
```

```
    """
    Lists the specified number of roles for the account.

    :param count: The number of roles to list.
    """
    try:
        roles = list(iam.roles.limit(count=count))
        for role in roles:
            logger.info("Role: %s", role.name)
    except ClientError:
        logger.exception("Couldn't list roles for the account.")
        raise
    else:
        return roles
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListRoles。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Lists IAM roles up to a specified count.
# @param count [Integer] the maximum number of roles to list.
# @return [Array<String>] the names of the roles.
def list_roles(count)
  role_names = []
  roles_counted = 0

  @iam_client.list_roles.each_page do |page|
    page.roles.each do |role|
      break if roles_counted >= count
```

```
        @logger.info("\t#{roles_counted + 1}: #{role.role_name}")
        role_names << role.role_name
        roles_counted += 1
      end
      break if roles_counted >= count
    end

    role_names
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Couldn't list roles for the account. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListRoles。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_roles(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListRolesOutput, SdkError<ListRolesError>> {
    let response = client
        .list_roles()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
```

```
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListRoles。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        oo_result = lo_iam->listroles( ).
        MESSAGE 'Retrieved role list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing roles.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListRoles。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3
```

```
public func listRoles() async throws -> [String] {
    var roleList: [String] = []

    // Use "Paginated" to get all the roles.
    // This lets the SDK handle the 'isTruncated' in "ListRolesOutput".
    let input = ListRolesInput()
    let pages = client.listRolesPaginated(input: input)

    do {
        for try await page in pages {
            guard let roles = page.roles else {
                print("Error: no roles returned.")
                continue
            }

            for role in roles {
                if let name = role.roleName {
                    roleList.append(name)
                }
            }
        }
    } catch {
        print("ERROR: listRoles:", dump(error))
        throw error
    }
    return roleList
}
```

- 有关 API 详细信息，请参阅 Amazon SDK for Swift API 参考中的 ListRoles。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListSAMLProviders** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListSAMLProviders。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
    /// <summary>
    /// List SAML authentication providers.
    /// </summary>
    /// <returns>A list of SAML providers.</returns>
    public async Task<List<SAMLProviderListEntry>> ListSAMLProvidersAsync()
    {
        var response = await _IAMService.ListSAMLProvidersAsync(new
ListSAMLProvidersRequest());
        return response.SAMLProviderList;
    }
```

- 有关 API 详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListSAMLProviders。

CLI

Amazon CLI

列出 Amazon 账户中的 SAML 提供者

此示例将检索在当前 Amazon 账户中创建的 SAML 2.0 提供者列表。

```
aws iam list-saml-providers
```

输出：

```
{
    "SAMLProviderList": [
```

```
        {
            "Arn": "arn:aws:iam::123456789012:saml-provider/SAML-ADFS",
            "ValidUntil": "2015-06-05T22:45:14Z",
            "CreateDate": "2015-06-05T22:45:14Z"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM SAML 身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListSAMLProviders。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
)

// AccountWrapper encapsulates AWS Identity and Access Management (IAM) account
 actions
// used in the examples.
// It contains an IAM service client that is used to perform account actions.
type AccountWrapper struct {
 IamClient *iam.Client
}



// ListSAMLProviders gets the SAML providers for the account.
```

```
func (wrapper AccountWrapper) ListSAMLProviders(ctx context.Context)
 ([]types.SAMLProviderListEntry, error) {
 var providers []types.SAMLProviderListEntry
 result, err := wrapper.IamClient.ListSAMLProviders(ctx,
 &iam.ListSAMLProvidersInput{})
 if err != nil {
  log.Printf("Couldn't list SAML providers. Here's why: %v\n", err)
 } else {
  providers = result.SAMLProviderList
 }
 return providers, err
}
```

- 有关 API 详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListSAMLProviders。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出 SAML 提供商。

```
import { ListSAMLProvidersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listSamlProviders = async () => {
  const command = new ListSAMLProvidersCommand({});

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListSAMLProviders。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listSAMLProviders()
    {
        return $this->iamClient->listSAMLProviders();
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListSAMLProviders。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将检索在当前 Amazon Web Services 账户 中创建的 SAML 2.0 提供者列表。其返回每个 SAML 提供者的 ARN、创建日期和到期日期。

```
Get-IAMSAMLProviderList
```

输出：

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
```

```
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListSAMLProviders。

Tools for PowerShell V5

示例 1：此示例将检索在当前 Amazon Web Services 账户 中创建的 SAML 2.0 提供者列表。其返回每个 SAML 提供者的 ARN、创建日期和到期日期。

```
Get-IAMSAMLProviderList
```

输出：

```
Arn                                                 CreateDate
   ValidUntil
---                                                 ----------
   ----------
arn:aws:iam::123456789012:saml-provider/SAMLADFS    12/23/2014 12:16:55 PM
   12/23/2114 12:16:54 PM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListSAMLProviders。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_saml_providers(count):
    """
    Lists the SAML providers for the account.

    :param count: The maximum number of providers to list.
    """
```

```
    try:
        found = 0
        for provider in iam.saml_providers.limit(count):
            logger.info("Got SAML provider %s.", provider.arn)
            found += 1
        if found == 0:
            logger.info("Your account has no SAML providers.")
    except ClientError:
        logger.exception("Couldn't list SAML providers.")
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 ListSAMLProviders。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class SamlProviderLister
  # Initializes the SamlProviderLister with IAM client and a logger.
  # @param iam_client [Aws::IAM::Client] The IAM client object.
  # @param logger [Logger] The logger object for logging output.
  def initialize(iam_client, logger = Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
  end

  # Lists up to a specified number of SAML providers for the account.
  # @param count [Integer] The maximum number of providers to list.
  # @return [Aws::IAM::Client::Response]
  def list_saml_providers(count)
    response = @iam_client.list_saml_providers
```

```
    response.saml_provider_list.take(count).each do |provider|
      @logger.info("\t#{provider.arn}")
    end
    response
  rescue Aws::Errors::ServiceError => e
    @logger.error("Couldn't list SAML providers. Here's why:")
    @logger.error("\t#{e.code}: #{e.message}")
    raise
  end
end
```

- 有关 API 详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListSAMLProviders。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_saml_providers(
    client: &Client,
) -> Result<ListSamlProvidersOutput, SdkError<ListSAMLProvidersError>> {
    let response = client.list_saml_providers().send().await?;

    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListSAMLProviders。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    oo_result = lo_iam->listsamlproviders( ).
    MESSAGE 'Retrieved SAML provider list.' TYPE 'I'.
  CATCH /aws1/cx_iamservicefailureex.
    MESSAGE 'Service failure when listing SAML providers.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListSAMLProviders。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListServerCertificates** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListServerCertificates`。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
bool AwsDoc::IAM::listServerCertificates(
        const Aws::Client::ClientConfiguration &clientConfig) {
```

```
    const Aws::String DATE_FORMAT = "%Y-%m-%d";

    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListServerCertificatesRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListServerCertificates(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list server certificates: " <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(55) << "Name" <<
                      std::setw(30) << "ID" << std::setw(80) << "Arn" <<
                      std::setw(14) << "UploadDate" << std::setw(14) <<
                      "ExpirationDate" << std::endl;
            header = true;
        }

        const auto &certificates =
                outcome.GetResult().GetServerCertificateMetadataList();

        for (const auto &certificate: certificates) {
            std::cout << std::left << std::setw(55) <<
                      certificate.GetServerCertificateName() << std::setw(30) <<
                      certificate.GetServerCertificateId() << std::setw(80) <<
                      certificate.GetArn() << std::setw(14) <<

certificate.GetUploadDate().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::setw(14) <<

certificate.GetExpiration().ToGmtString(DATE_FORMAT.c_str()) <<
                      std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
```

```
        }
    }

    return true;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 ListServerCertificates。

CLI

Amazon CLI

列出 Amazon 账户中的服务器证书

以下 `list-server-certificates` 命令将列出 Amazon 账户中存储并可供使用的所有服务器证书。

```
aws iam list-server-certificates
```

输出：

```
{
    "ServerCertificateMetadataList": [
        {
            "Path": "/",
            "ServerCertificateName": "myUpdatedServerCertificate",
            "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/
myUpdatedServerCertificate",
            "UploadDate": "2019-04-22T21:13:44+00:00",
            "Expiration": "2019-10-15T22:23:16+00:00"
        },
        {
            "Path": "/cloudfront/",
            "ServerCertificateName": "MyTestCert",
            "ServerCertificateId": "ASCAEXAMPLE456EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyTestCert",
            "UploadDate": "2015-04-21T18:14:16+00:00",
            "Expiration": "2018-01-14T17:52:36+00:00"
```

```
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 中管理服务器证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListServerCertificates。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出证书。

```
import { ListServerCertificatesCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 * A generator function that handles paginated results.
 * The AWS SDK for JavaScript (v3) provides {@link https://docs.aws.amazon.com/
AWSJavaScriptSDK/v3/latest/index.html#paginators | paginator} functions to
 simplify this.
 *
 */
export async function* listServerCertificates() {
  const command = new ListServerCertificatesCommand({});
  let response = await client.send(command);

  while (response.ServerCertificateMetadataList?.length) {
    for await (const cert of response.ServerCertificateMetadataList) {
      yield cert;
    }

    if (response.IsTruncated) {
      response = await client.send(new ListServerCertificatesCommand({}));
```

```
    } else {
      break;
    }
  }
}
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-listing。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListServerCertificates。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

iam.listServerCertificates({}, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-listing。

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListServerCertificates。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索已上传到当前 Amazon Web Services 账户 的服务器证书列表。

```
Get-IAMServerCertificateList
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListServerCertificates。

Tools for PowerShell V5

示例 1：此示例检索已上传到当前 Amazon Web Services 账户 的服务器证书列表。

```
Get-IAMServerCertificateList
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/Org1/Org2/
MyServerCertificate
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /Org1/Org2/
ServerCertificateId   : ASCAJIFEXAMPLE17HQZYW
ServerCertificateName : MyServerCertificate
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListServerCertificates。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates
```

```
    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
 #{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
 '#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 ListServerCertificates。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListSigningCertificates** 与 CLI 配合使用

以下代码示例演示如何使用 ListSigningCertificates。

CLI

Amazon CLI

列出 IAM 用户的签名证书

以下 list-signing-certificates 命令列出名为 Bob 的 IAM 用户的签名证书。

```
aws iam list-signing-certificates \
    --user-name Bob
```

输出：

```
{
    "Certificates": [
        {
            "UserName": "Bob",
            "Status": "Inactive",
            "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-
body>-----END CERTIFICATE-----",
            "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
            "UploadDate": "2013-06-06T21:40:08Z"
        }
    ]
}
```

有关更多信息，请参阅《Amazon EC2 用户指南》中的管理签名证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListSigningCertificates。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索与名为 **Bob** 的用户关联的签名证书的相关详细信息。

```
Get-IAMSigningCertificate -UserName Bob
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                   YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                   21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                   rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                   nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                   NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                   -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListSigningCertificates。

Tools for PowerShell V5

示例 1：此示例检索与名为 **Bob** 的用户关联的签名证书的相关详细信息。

```
Get-IAMSigningCertificate -UserName Bob
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                   YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                   21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                   rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                   nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                   NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                   -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListSigningCertificates。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListUserPolicies** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListUserPolicies`。

CLI

Amazon CLI

列出 IAM 用户的策略

以下 `list-user-policies` 命令列出附加到名为 Bob 的 IAM 用户的策略。

```
aws iam list-user-policies \
    --user-name Bob
```

输出：

```
{
    "PolicyNames": [
        "ExamplePolicy",
        "TestPolicy"
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 账户中创建 IAM 用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListUserPolicies。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// ListUserPolicies lists the inline policies for the specified user.
func (wrapper UserWrapper) ListUserPolicies(ctx context.Context, userName string)
 ([]string, error) {
 var policies []string
 result, err := wrapper.IamClient.ListUserPolicies(ctx,
 &iam.ListUserPoliciesInput{
  UserName: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't list policies for user %v. Here's why: %v\n", userName,
 err)
```

```
 } else {
  policies = result.PolicyNames
 }
 return policies, err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 ListUserPolicies。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中的内联策略名称列表。

```
Get-IAMUserPolicyList -UserName David
```

输出：

```
Davids_IAM_Admin_Policy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListUserPolicies。

Tools for PowerShell V5

示例 1：此示例检索嵌入在名为 **David** 的 IAM 用户中的内联策略名称列表。

```
Get-IAMUserPolicyList -UserName David
```

输出：

```
Davids_IAM_Admin_Policy
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListUserPolicies。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListUserTags** 与 CLI 配合使用

以下代码示例演示如何使用 ListUserTags。

CLI

Amazon CLI

列出附加到用户的标签

以下 list-user-tags 命令检索与指定 IAM 用户关联的标签列表。

```
aws iam list-user-tags \
    --user-name alice
```

输出：

```
{
    "Tags": [
        {
            "Key": "Department",
            "Value": "Accounting"
        },
        {
            "Key": "DeptID",
            "Value": "12345"
        }
    ],
    "IsTruncated": false
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListUserTags。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例获取与用户关联的标签。

```
Get-IAMUserTagList -UserName joe
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListUserTags。

Tools for PowerShell V5

示例 1：此示例获取与用户关联的标签。

```
Get-IAMUserTagList -UserName joe
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListUserTags。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListUsers** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ListUsers。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 创建只读和读写用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
/// <summary>
/// List IAM users.
/// </summary>
/// <returns>A list of IAM users.</returns>
```

```
    public async Task<List<User>> ListUsersAsync()
    {
        var listUsersPaginator = _IAMService.Paginators.ListUsers(new
 ListUsersRequest());
        var users = new List<User>();

        await foreach (var response in listUsersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考中的 ListUsers。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function iam_list_users
#
# List the IAM users in the account.
#
```

```
# Returns:
#       The list of users names
#    And:
#       0 - If the user already exists.
#       1 - If the user doesn't exist.
###############################################################################
function iam_list_users() {
  local option OPTARG # Required to use getopts command in a function.
  local error_code
  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_list_users"
    echo "Lists the AWS Identity and Access Management (IAM) user in the
 account."
    echo ""
  }

  # Retrieve the calling parameters.
  while getopts "h" option; do
    case "${option}" in
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done
  export OPTIND=1

  local response

  response=$(aws iam list-users \
    --output text \
    --query "Users[].UserName")
  error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports list-users operation failed.$response"
    return 1
```

```
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 ListUsers。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::listUsers(const Aws::Client::ClientConfiguration &clientConfig)
 {
    const Aws::String DATE_FORMAT = "%Y-%m-%d";
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::ListUsersRequest request;

    bool done = false;
    bool header = false;
    while (!done) {
        auto outcome = iam.ListUsers(request);
        if (!outcome.IsSuccess()) {
            std::cerr << "Failed to list iam users:" <<
                      outcome.GetError().GetMessage() << std::endl;
            return false;
        }

        if (!header) {
            std::cout << std::left << std::setw(32) << "Name" <<
                      std::setw(30) << "ID" << std::setw(64) << "Arn" <<
                      std::setw(20) << "CreateDate" << std::endl;
            header = true;
        }
```

```
        const auto &users = outcome.GetResult().GetUsers();
        for (const auto &user: users) {
            std::cout << std::left << std::setw(32) << user.GetUserName() <<
                      std::setw(30) << user.GetUserId() << std::setw(64) <<
                      user.GetArn() << std::setw(20) <<
                      user.GetCreateDate().ToGmtString(DATE_FORMAT.c_str())
                      << std::endl;
        }

        if (outcome.GetResult().GetIsTruncated()) {
            request.SetMarker(outcome.GetResult().GetMarker());
        }
        else {
            done = true;
        }
    }

    return true;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考中的 ListUsers。

CLI

Amazon CLI

列出 IAM 用户

以下 `list-users` 命令将列出当前账户中的 IAM 用户。

```
aws iam list-users
```

输出：

```
{
    "Users": [
        {
            "UserName": "Adele",
            "Path": "/",
            "CreateDate": "2013-03-07T05:14:48Z",
            "UserId": "AKIAI44QH8DHBEXAMPLE",
```

```
            "Arn": "arn:aws:iam::123456789012:user/Adele"
        },
        {
            "UserName": "Bob",
            "Path": "/",
            "CreateDate": "2012-09-21T23:03:13Z",
            "UserId": "AKIAIOSFODNN7EXAMPLE",
            "Arn": "arn:aws:iam::123456789012:user/Bob"
        }
    ]
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的列出 IAM 用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListUsers。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
```

```
 IamClient *iam.Client
}



// ListUsers gets up to maxUsers number of users.
func (wrapper UserWrapper) ListUsers(ctx context.Context, maxUsers int32)
 ([]types.User, error) {
 var users []types.User
 result, err := wrapper.IamClient.ListUsers(ctx, &iam.ListUsersInput{
  MaxItems: aws.Int32(maxUsers),
 })
 if err != nil {
  log.Printf("Couldn't list users. Here's why: %v\n", err)
 } else {
  users = result.Users
 }
 return users, err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考中的 ListUsers。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.AttachedPermissionsBoundary;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.ListUsersRequest;
import software.amazon.awssdk.services.iam.model.ListUsersResponse;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.User;
```

```
/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        listAllUsers(iam);
        System.out.println("Done");
        iam.close();
    }

    public static void listAllUsers(IamClient iam) {
        try {
            boolean done = false;
            String newMarker = null;
            while (!done) {
                ListUsersResponse response;
                if (newMarker == null) {
                    ListUsersRequest request =
 ListUsersRequest.builder().build();
                    response = iam.listUsers(request);
                } else {
                    ListUsersRequest request = ListUsersRequest.builder()
                            .marker(newMarker)
                            .build();

                    response = iam.listUsers(request);
                }

                for (User user : response.users()) {
                    System.out.format("\n Retrieved user %s", user.userName());
                    AttachedPermissionsBoundary permissionsBoundary =
 user.permissionsBoundary();
                    if (permissionsBoundary != null)
```

```
                        System.out.format("\n Permissions boundary details %s",

 permissionsBoundary.permissionsBoundaryTypeAsString());
                }

                if (!response.isTruncated()) {
                    done = true;
                } else {
                    newMarker = response.marker();
                }
            }

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考中的 ListUsers。

JavaScript

适用于 JavaScript 的开发工具包（v3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出用户。

```
import { ListUsersCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

export const listUsers = async () => {
  const command = new ListUsersCommand({ MaxItems: 10 });

  const response = await client.send(command);
```

```
  for (const { UserName, CreateDate } of response.Users) {
    console.log(`${UserName} created on: ${CreateDate}`);
  }
  return response;
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-listing-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListUsers。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  MaxItems: 10,
};

iam.listUsers(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    var users = data.Users || [];
    users.forEach(function (user) {
      console.log("User " + user.UserName + " created", user.CreateDate);
    });
```

```
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-listing-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 ListUsers。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun listAllUsers() {
    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        val response = iamClient.listUsers(ListUsersRequest { })
        response.users?.forEach { user ->
            println("Retrieved user ${user.userName}")
            val permissionsBoundary = user.permissionsBoundary
            if (permissionsBoundary != null) {
                println("Permissions boundary details
 ${permissionsBoundary.permissionsBoundaryType}")
            }
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 ListUsers。

PHP

适用于 PHP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
$uuid = uniqid();
$service = new IAMService();

    public function listUsers($pathPrefix = "", $marker = "", $maxItems = 0)
    {
        $listUsersArguments = [];
        if ($pathPrefix) {
            $listUsersArguments["PathPrefix"] = $pathPrefix;
        }
        if ($marker) {
            $listUsersArguments["Marker"] = $marker;
        }
        if ($maxItems) {
            $listUsersArguments["MaxItems"] = $maxItems;
        }

        return $this->iamClient->listUsers($listUsersArguments);
    }
```

- 有关 API 的详细信息，请参阅 适用于 PHP 的 Amazon SDK API 参考中的 ListUsers。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索当前 Amazon Web Services 账户 中的用户集合。

```
Get-IAMUserList
```

输出：

```
    Arn              : arn:aws:iam::123456789012:user/Administrator
    CreateDate       : 10/16/2014 9:03:09 AM
    PasswordLastUsed : 3/4/2015 12:12:33 PM
    Path             : /
    UserId           : 7K3GJEANSKZF2EXAMPLE1
    UserName         : Administrator

    Arn              : arn:aws:iam::123456789012:user/Bob
    CreateDate       : 4/6/2015 12:54:42 PM
    PasswordLastUsed : 1/1/0001 12:00:00 AM
    Path             : /
    UserId           : L3EWNONDOM3YUEXAMPLE2
    UserName         : bab

    Arn              : arn:aws:iam::123456789012:user/David
    CreateDate       : 12/10/2014 3:39:27 PM
    PasswordLastUsed : 3/19/2015 8:44:04 AM
    Path             : /
    UserId           : Y4FKWQCXTA52QEXAMPLE3
    UserName         : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListUsers。

Tools for PowerShell V5

示例 1：此示例检索当前 Amazon Web Services 账户 中的用户集合。

```
Get-IAMUserList
```

输出：

```
    Arn              : arn:aws:iam::123456789012:user/Administrator
    CreateDate       : 10/16/2014 9:03:09 AM
    PasswordLastUsed : 3/4/2015 12:12:33 PM
    Path             : /
    UserId           : 7K3GJEANSKZF2EXAMPLE1
    UserName         : Administrator

    Arn              : arn:aws:iam::123456789012:user/Bob
    CreateDate       : 4/6/2015 12:54:42 PM
    PasswordLastUsed : 1/1/0001 12:00:00 AM
    Path             : /
```

```
      UserId           : L3EWNONDOM3YUEXAMPLE2
      UserName         : bab

      Arn              : arn:aws:iam::123456789012:user/David
      CreateDate       : 12/10/2014 3:39:27 PM
      PasswordLastUsed : 3/19/2015 8:44:04 AM
      Path             : /
      UserId           : Y4FKWQCXTA52QEXAMPLE3
      UserName         : David
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 ListUsers。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 ListUsers。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Lists all users in the AWS account
#
# @return [Array<Aws::IAM::Types::User>] An array of user objects
def list_users
  users = []
  @iam_client.list_users.each_page do |page|
    page.users.each do |user|
      users << user
    end
  end
  users
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Error listing users: #{e.message}")
  []
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考中的 ListUsers。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
pub async fn list_users(
    client: &iamClient,
    path_prefix: Option<String>,
    marker: Option<String>,
    max_items: Option<i32>,
) -> Result<ListUsersOutput, SdkError<ListUsersError>> {
    let response = client
        .list_users()
        .set_path_prefix(path_prefix)
        .set_marker(marker)
        .set_max_items(max_items)
        .send()
        .await?;
    Ok(response)
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 ListUsers。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
```

```
        oo_result = lo_iam->listusers( ).
        MESSAGE 'Retrieved user list.' TYPE 'I'.
      CATCH /aws1/cx_iamservicefailureex.
        MESSAGE 'Service failure when listing users.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 ListUsers。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    public func listUsers() async throws -> [MyUserRecord] {
        var userList: [MyUserRecord] = []

        // Use "Paginated" to get all the users.
        // This lets the SDK handle the 'isTruncated' in "ListUsersOutput".
        let input = ListUsersInput()
        let output = client.listUsersPaginated(input: input)

        do {
            for try await page in output {
                guard let users = page.users else {
                    continue
                }
                for user in users {
                    if let id = user.userId, let name = user.userName {
                        userList.append(MyUserRecord(id: id, name: name))
                    }
```

```
                }
            }
        }
        catch {
            print("ERROR: listUsers:", dump(error))
            throw error
        }
       return userList
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Swift API 参考中的 ListUsers。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ListVirtualMfaDevices** 与 CLI 配合使用

以下代码示例演示如何使用 `ListVirtualMfaDevices`。

CLI

Amazon CLI

列出虚拟 MFA 设备

以下 `list-virtual-mfa-devices` 命令列出已为当前账户配置的虚拟 MFA 设备。

```
aws iam list-virtual-mfa-devices
```

输出：

```
{
    "VirtualMFADevices": [
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/ExampleMFADevice"
        },
        {
            "SerialNumber": "arn:aws:iam::123456789012:mfa/Fred"
        }
    ]
```

```
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的启用虚拟多重身份验证（MFA）设备。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ListVirtualMfaDevices。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例检索分配给 Amazon 账户中用户的虚拟 MFA 设备集合。每个设备的 **User** 属性都是一个对象，其中包含设备分配到的 IAM 用户的详细信息。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

输出：

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ListVirtualMfaDevices。

Tools for PowerShell V5

示例 1：此示例检索分配给 Amazon 账户中用户的虚拟 MFA 设备集合。每个设备的 **User** 属性都是一个对象，其中包含设备分配到的 IAM 用户的详细信息。

```
Get-IAMVirtualMFADevice -AssignmentStatus Assigned
```

输出：

```
Base32StringSeed :
EnableDate       : 4/13/2015 12:03:42 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/David
User             : Amazon.IdentityManagement.Model.User

Base32StringSeed :
EnableDate       : 4/13/2015 12:06:41 PM
QRCodePNG        :
SerialNumber     : arn:aws:iam::123456789012:mfa/root-account-mfa-device
User             : Amazon.IdentityManagement.Model.User
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ListVirtualMfaDevices。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutGroupPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 PutGroupPolicy。

CLI

Amazon CLI

为组添加策略

以下 put-group-policy 命令可将策略添加到名为 Admins 的 IAM 组。

```
aws iam put-group-policy \
    --group-name Admins \
    --policy-document file://AdminPolicy.json \
    --policy-name AdminRoot
```

此命令不生成任何输出。

该策略在 AdminPolicy.json 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM policy。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 PutGroupPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **AppTesterPolicy** 的内联策略并将其嵌入到 IAM 组 **AppTesters** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **apptesterpolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 PutGroupPolicy。

Tools for PowerShell V5

示例 1：此示例创建一个名为 **AppTesterPolicy** 的内联策略并将其嵌入到 IAM 组 **AppTesters** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **apptesterpolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMGroupPolicy -GroupName AppTesters -PolicyName AppTesterPolicy -
PolicyDocument (Get-Content -Raw apptesterpolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 PutGroupPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutRolePermissionsBoundary** 与 CLI 配合使用

以下代码示例演示如何使用 PutRolePermissionsBoundary。

CLI

Amazon CLI

示例 1：将基于自定义策略的权限边界应用于 IAM 角色

以下 put-role-permissions-boundary 示例应用名为 intern-boundary 的自定义策略作为指定 IAM 角色的权限边界。

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --role-name lambda-application-role
```

此命令不生成任何输出。

示例 2：将基于 Amazon 托管策略的权限边界应用于 IAM 角色

以下 put-role-permissions-boundary 示例应用 Amazon 托管 PowerUserAccess 策略作为指定 IAM 角色的权限边界。

```
aws iam put-role-permissions-boundary \
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --role-name x-account-admin
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 PutRolePermissionsBoundary。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例展示如何设置 IAM 角色的权限边界。您可以将 Amazon 托管策略或自定义策略设置为权限边界。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 PutRolePermissionsBoundary。

Tools for PowerShell V5

示例 1：此示例展示如何设置 IAM 角色的权限边界。您可以将 Amazon 托管策略或自定义策略设置为权限边界。

```
Set-IAMRolePermissionsBoundary -RoleName MyRoleName -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 PutRolePermissionsBoundary。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutRolePolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutRolePolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- Redshift Serverless 入门
- Amazon Redshift 预置集群入门
- Config 入门
- IoT Device Defender 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Update the inline policy document embedded in a role.
    /// </summary>
    /// <param name="policyName">The name of the policy to embed.</param>
    /// <param name="roleName">The name of the role to update.</param>
```

```
    /// <param name="policyDocument">The policy document that defines the role.</
param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> PutRolePolicyAsync(string policyName, string
 roleName, string policyDocument)
    {
        var request = new PutRolePolicyRequest
        {
            PolicyName = policyName,
            RoleName = roleName,
            PolicyDocument = policyDocument
        };

        var response = await _IAMService.PutRolePolicyAsync(request);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 PutRolePolicy。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::putRolePolicy(
        const Aws::String &roleName,
        const Aws::String &policyName,
        const Aws::String &policyDocument,
        const Aws::Client::ClientConfiguration &clientConfig) {
    Aws::IAM::IAMClient iamClient(clientConfig);
    Aws::IAM::Model::PutRolePolicyRequest request;

    request.SetRoleName(roleName);
```

```
    request.SetPolicyName(policyName);
    request.SetPolicyDocument(policyDocument);

    Aws::IAM::Model::PutRolePolicyOutcome outcome =
 iamClient.PutRolePolicy(request);
    if (!outcome.IsSuccess()) {
        std::cerr << "Error putting policy on role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Successfully put the role policy." << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 PutRolePolicy。

CLI

Amazon CLI

将权限策略附加到 IAM 角色

以下 `put-role-policy` 命令可将权限策略附加到名为 `Test-Role` 的角色。

```
aws iam put-role-policy \
    --role-name Test-Role \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

此命令不生成任何输出。

该策略在 AdminPolicy.json 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）

要将信任策略附加到角色，请使用 `update-assume-role-policy` 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 PutRolePolicy。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { PutRolePolicyCommand, IAMClient } from "@aws-sdk/client-iam";

const examplePolicyDocument = JSON.stringify({
  Version: "2012-10-17",
  Statement: [
    {
      Sid: "VisualEditor0",
      Effect: "Allow",
      Action: [
        "s3:ListBucketMultipartUploads",
        "s3:ListBucketVersions",
        "s3:ListBucket",
        "s3:ListMultipartUploadParts",
      ],
      Resource: "arn:aws:s3:::amzn-s3-demo-bucket",
    },
    {
      Sid: "VisualEditor1",
      Effect: "Allow",
      Action: [
        "s3:ListStorageLensConfigurations",
        "s3:ListAccessPointsForObjectLambda",
        "s3:ListAllMyBuckets",
        "s3:ListAccessPoints",
        "s3:ListJobs",
        "s3:ListMultiRegionAccessPoints",
      ],
      Resource: "*",
    },
  ],
});
```

```
const client = new IAMClient({});

/**
 *
 * @param {string} roleName
 * @param {string} policyName
 * @param {string} policyDocument
 */
export const putRolePolicy = async (roleName, policyName, policyDocument) => {
  const command = new PutRolePolicyCommand({
    RoleName: roleName,
    PolicyName: policyName,
    PolicyDocument: policyDocument,
  });

  const response = await client.send(command);
  console.log(response);
  return response;
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 PutRolePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **FedTesterRolePolicy** 的内联策略并将其嵌入到 IAM 角色 **FedTesterRole** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **FedTesterPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 PutRolePolicy。

Tools for PowerShell V5

示例 1：此示例创建一个名为 **FedTesterRolePolicy** 的内联策略并将其嵌入到 IAM 角色 **FedTesterRole** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **FedTesterPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMRolePolicy -RoleName FedTesterRole -PolicyName FedTesterRolePolicy -
PolicyDocument (Get-Content -Raw FedTesterPolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 PutRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutUserPermissionsBoundary** 与 CLI 配合使用

以下代码示例演示如何使用 PutUserPermissionsBoundary。

CLI

Amazon CLI

示例 1：将基于自定义策略的权限边界应用于 IAM 用户

以下 put-user-permissions-boundary 示例应用名为 intern-boundary 的自定义策略作为指定 IAM 用户的权限边界。

```
aws iam put-user-permissions-boundary \
    --permissions-boundary arn:aws:iam::123456789012:policy/intern-boundary \
    --user-name intern
```

此命令不生成任何输出。

示例 2：将基于 Amazon 托管策略的权限边界应用于 IAM 用户

以下 put-user-permissions-boundary 示例应用名为 PowerUserAccess 的 Amazon 托管策略作为指定 IAM 用户的权限边界。

```
aws iam put-user-permissions-boundary \
```

```
    --permissions-boundary arn:aws:iam::aws:policy/PowerUserAccess \
    --user-name developer
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的添加和移除 IAM 身份权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 PutUserPermissionsBoundary。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例展示如何设置用户的权限边界。您可以将 Amazon 托管策略或自定义策略设置为权限边界。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 PutUserPermissionsBoundary。

Tools for PowerShell V5

示例 1：此示例展示如何设置用户的权限边界。您可以将 Amazon 托管策略或自定义策略设置为权限边界。

```
Set-IAMUserPermissionsBoundary -UserName joe -PermissionsBoundary
 arn:aws:iam::123456789012:policy/intern-boundary
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 PutUserPermissionsBoundary。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **PutUserPolicy** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 PutUserPolicy。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 了解基本功能

CLI

Amazon CLI

将策略附加到 IAM 用户

以下 put-user-policy 命令可将策略附加到名为 Bob 的 IAM 用户。

```
aws iam put-user-policy \
    --user-name Bob \
    --policy-name ExamplePolicy \
    --policy-document file://AdminPolicy.json
```

此命令不生成任何输出。

该策略在 AdminPolicy.json 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）

有关更多信息，请参阅《Amazon IAM 用户指南》中的添加和删除 IAM 身份权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 PutUserPolicy。

Go

适用于 Go 的 SDK V2

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "encoding/json"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/iam"
```

```
 "github.com/aws/aws-sdk-go-v2/service/iam/types"
 "github.com/aws/smithy-go"
)

// UserWrapper encapsulates user actions used in the examples.
// It contains an IAM service client that is used to perform user actions.
type UserWrapper struct {
 IamClient *iam.Client
}



// CreateUserPolicy adds an inline policy to a user. This example creates a
 policy that
// grants a list of actions on a specified role.
// PolicyDocument shows how to work with a policy document as a data structure
 and
// serialize it to JSON by using Go's JSON marshaler.
func (wrapper UserWrapper) CreateUserPolicy(ctx context.Context, userName string,
 policyName string, actions []string,
 roleArn string) error {
 policyDoc := PolicyDocument{
  Version: "2012-10-17",
  Statement: []PolicyStatement{{
   Effect:   "Allow",
   Action:   actions,
   Resource: aws.String(roleArn),
  }},
 }
 policyBytes, err := json.Marshal(policyDoc)
 if err != nil {
  log.Printf("Couldn't create policy document for %v. Here's why: %v\n", roleArn,
 err)
  return err
 }
 _, err = wrapper.IamClient.PutUserPolicy(ctx, &iam.PutUserPolicyInput{
  PolicyDocument: aws.String(string(policyBytes)),
  PolicyName:     aws.String(policyName),
  UserName:       aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't create policy for user %v. Here's why: %v\n", userName,
 err)
 }
```

```
 return err
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API Reference》中的 PutUserPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例创建一个名为 **EC2AccessPolicy** 的内联策略并将其嵌入到 IAM 用户 **Bob** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **EC2AccessPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 PutUserPolicy。

Tools for PowerShell V5

示例 1：此示例创建一个名为 **EC2AccessPolicy** 的内联策略并将其嵌入到 IAM 用户 **Bob** 中。如果已经存在同名的内联策略，则该策略将被覆盖。JSON 策略内容来自文件 **EC2AccessPolicy.json**。请注意，必须使用 **-Raw** 参数才能成功处理 JSON 文件的内容。

```
Write-IAMUserPolicy -UserName Bob -PolicyName EC2AccessPolicy -PolicyDocument
 (Get-Content -Raw EC2AccessPolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 PutUserPolicy。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Creates an inline policy for a specified user.
# @param username [String] The name of the IAM user.
# @param policy_name [String] The name of the policy to create.
# @param policy_document [String] The JSON policy document.
# @return [Boolean]
def create_user_policy(username, policy_name, policy_document)
  @iam_client.put_user_policy({
                                user_name: username,
                                policy_name: policy_name,
                                policy_document: policy_document
                              })
  @logger.info("Policy #{policy_name} created for user #{username}.")
  true
rescue Aws::IAM::Errors::ServiceError => e
  @logger.error("Couldn't create policy #{policy_name} for user #{username}.
Here's why:")
  @logger.error("\t#{e.code}: #{e.message}")
  false
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 PutUserPolicy。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSIAM
import AWSS3


    func putUserPolicy(policyDocument: String, policyName: String, user:
 IAMClientTypes.User) async throws {
        let input = PutUserPolicyInput(
            policyDocument: policyDocument,
            policyName: policyName,
            userName: user.userName
        )
        do {
            _ = try await iamClient.putUserPolicy(input: input)
        } catch {
            print("ERROR: putUserPolicy:", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 PutUserPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 `RemoveClientIdFromOpenIdConnectProvider` 与 CLI 配合使用

以下代码示例演示如何使用 `RemoveClientIdFromOpenIdConnectProvider`。

CLI

Amazon CLI

从为指定的 IAM OpenID Connect 提供者注册的客户端 ID 列表中删除指定的客户端 ID

此示例从与 ARN 为 `arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com` 的 IAM OIDC 提供者关联的客户端 ID 列表中删除客户端 ID `My-TestApp-3`。

```
aws iam remove-client-id-from-open-id-connect-provider
    --client-id My-TestApp-3 \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的 创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 RemoveClientIdFromOpenIdConnectProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从与 ARN 为 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 IAM OIDC 提供者关联的客户端 ID 列表中删除客户端 ID **My-TestApp-3**。

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 RemoveClientIdFromOpenIdConnectProvider。

Tools for PowerShell V5

示例 1：此示例从与 ARN 为 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 IAM OIDC 提供者关联的客户端 ID 列表中删除客户端 ID **My-TestApp-3**。

```
Remove-IAMClientIDFromOpenIDConnectProvider -ClientID My-TestApp-3
 -OpenIDConnectProviderArn arn:aws:iam::123456789012:oidc-provider/
example.oidcprovider.com
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 RemoveClientIdFromOpenIdConnectProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **RemoveRoleFromInstanceProfile** 与 CLI 配合使用

以下代码示例演示如何使用 RemoveRoleFromInstanceProfile。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- Amazon MSK 入门
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试

CLI

Amazon CLI

从实例配置文件中删除角色

以下 remove-role-from-instance-profile 命令可将名为 Test-Role 的角色从名为 ExampleInstanceProfile 的实例配置文件中删除。

```
aws iam remove-role-from-instance-profile \
    --instance-profile-name ExampleInstanceProfile \
    --role-name Test-Role
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的使用实例配置文件。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 RemoveRoleFromInstanceProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从名为 **MyNewRole** 的 EC2 实例配置文件中删除名为 **MyNewRole** 的角色。在 IAM 控制台中创建的实例配置文件始终与角色同名，如本示例所示。如果您在 API 或 CLI 中创建实例配置文件，则其可以有不同的名称。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 RemoveRoleFromInstanceProfile。

Tools for PowerShell V5

示例 1：此示例从名为 **MyNewRole** 的 EC2 实例配置文件中删除名为 **MyNewRole** 的角色。在 IAM 控制台中创建的实例配置文件始终与角色同名，如本示例所示。如果您在 API 或 CLI 中创建实例配置文件，则其可以有不同的名称。

```
Remove-IAMRoleFromInstanceProfile -InstanceProfileName MyNewRole -RoleName
 MyNewRole -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 RemoveRoleFromInstanceProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **RemoveUserFromGroup** 与 CLI 配合使用

以下代码示例演示如何使用 RemoveUserFromGroup。

CLI

Amazon CLI

从 IAM 组中删除用户

以下 remove-user-from-group 命令可将名为 Bob 的用户从名为 Admins 的 IAM 组中删除。

```
aws iam remove-user-from-group \
    --user-name Bob \
    --group-name Admins
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 用户组中添加和删除用户。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 RemoveUserFromGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从组 **Testers** 中删除 IAM 用户 **Bob**。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

示例 2：此示例查找 IAM 用户 **Theresa** 所属的所有组，然后从这些组中删除 **Theresa**。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

示例 3：此示例显示了从 **Testers** 组中删除 IAM 用户 **Bob** 的另一种方法。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 RemoveUserFromGroup。

Tools for PowerShell V5

示例 1：此示例从组 **Testers** 中删除 IAM 用户 **Bob**。

```
Remove-IAMUserFromGroup -GroupName Testers -UserName Bob
```

示例 2：此示例查找 IAM 用户 **Theresa** 所属的所有组，然后从这些组中删除 **Theresa**。

```
$groups = Get-IAMGroupForUser -UserName Theresa
foreach ($group in $groups) { Remove-IAMUserFromGroup -GroupName $group.GroupName
 -UserName Theresa -Force }
```

示例 3：此示例显示了从 **Testers** 组中删除 IAM 用户 **Bob** 的另一种方法。

```
Get-IAMGroupForUser -UserName Bob | Remove-IAMUserFromGroup -UserName Bob -
GroupName Testers -Force
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 RemoveUserFromGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **ResyncMfaDevice** 与 CLI 配合使用

以下代码示例演示如何使用 ResyncMfaDevice。

CLI

Amazon CLI

重新同步 MFA 设备

以下 resync-mfa-device 示例将与 IAM 用户 Bob 关联且 ARN 为 arn:aws:iam::123456789012:mfa/BobsMFADevice 的 MFA 设备与提供这两个身份验证码的身份验证器程序同步。

```
aws iam resync-mfa-device \
    --user-name Bob \
    --serial-number arn:aws:iam::210987654321:mfa/BobsMFADevice \
    --authentication-code1 123456 \
    --authentication-code2 987654
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 Amazon 中使用多重身份验证（MFA）。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 ResyncMfaDevice。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将与 IAM 用户 **Bob** 关联且 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 MFA 设备与提供这两个身份验证码的身份验证器程序同步。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

示例 2：此示例将与 IAM 用户 **Theresa** 关联的 IAM MFA 设备与具有序列号 **ABCD12345678** 并提供两个身份验证码的物理设备同步。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 ResyncMfaDevice。

Tools for PowerShell V5

示例 1：此示例将与 IAM 用户 **Bob** 关联且 ARN 为 **arn:aws:iam::123456789012:mfa/bob** 的 MFA 设备与提供这两个身份验证码的身份验证器程序同步。

```
Sync-IAMMFADevice -SerialNumber arn:aws:iam::123456789012:mfa/theresa -
AuthenticationCode1 123456 -AuthenticationCode2 987654 -UserName Bob
```

示例 2：此示例将与 IAM 用户 **Theresa** 关联的 IAM MFA 设备与具有序列号 **ABCD12345678** 并提供两个身份验证码的物理设备同步。

```
Sync-IAMMFADevice -SerialNumber ABCD12345678 -AuthenticationCode1 123456 -
AuthenticationCode2 987654 -UserName Theresa
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 ResyncMfaDevice。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SetDefaultPolicyVersion** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 SetDefaultPolicyVersion。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 管理策略
- 回滚策略版本

CLI

Amazon CLI

将指定策略的指定版本设置为策略的默认版本。

此示例将 ARN 为 arn:aws:iam::123456789012:policy/MyPolicy 的策略的 v2 版本设置为默认活动版本。

```
aws iam set-default-policy-version \
    --policy-arn arn:aws:iam::123456789012:policy/MyPolicy \
    --version-id v2
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的 IAM 中的策略和权限。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 SetDefaultPolicyVersion。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy** 的策略的 **v2** 版本设置为默认活动版本。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 SetDefaultPolicyVersion。

Tools for PowerShell V5

示例 1：此示例将 ARN 为 **arn:aws:iam::123456789012:policy/MyPolicy** 的策略的 **v2** 版本设置为默认活动版本。

```
Set-IAMDefaultPolicyVersion -PolicyArn arn:aws:iam::123456789012:policy/MyPolicy
 -VersionId v2
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 SetDefaultPolicyVersion。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    TRY.
        lo_iam->setdefaultpolicyversion(
          iv_policyarn = iv_policy_arn
          iv_versionid = iv_version_id ).
        MESSAGE 'Default policy version set successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'Policy or version does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iaminvalidinputex.
        MESSAGE 'Invalid input provided.' TYPE 'E'.
      CATCH /aws1/cx_iamlimitexceededex.
        MESSAGE 'Limit exceeded.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 SetDefaultPolicyVersion。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **TagRole** 与 CLI 配合使用

以下代码示例演示如何使用 TagRole。

CLI

Amazon CLI

为角色添加标签

以下 tag-role 命令为指定角色添加带有部门名称的标签。

```
aws iam tag-role --role-name my-role \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 TagRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在身份管理服务中为角色添加标签

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 TagRole。

Tools for PowerShell V5

示例 1：此示例在身份管理服务中为角色添加标签

```
Add-IAMRoleTag -RoleName AdminRoleacess -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 TagRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **TagUser** 与 CLI 配合使用

以下代码示例演示如何使用 TagUser。

CLI

Amazon CLI

为用户添加标签

以下 tag-user 命令为指定用户添加带有相关部门的标签。

```
aws iam tag-user \
    --user-name alice \
    --tags '{"Key": "Department", "Value": "Accounting"}'
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 TagUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例在身份管理服务中为用户添加标签

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 TagUser。

Tools for PowerShell V5

示例 1：此示例在身份管理服务中为用户添加标签

```
Add-IAMUserTag -UserName joe -Tag @{ Key = 'abac'; Value = 'testing'}
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 TagUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UntagRole** 与 CLI 配合使用

以下代码示例演示如何使用 UntagRole。

CLI

Amazon CLI

删除角色的标签

以下 untag-role 命令从指定角色中移除带有键名称“Department”的任何标签。

```
aws iam untag-role \
    --role-name my-role \
    --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UntagRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从名为“MyRoleName”的角色中删除标签键为“abac”的标签。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UntagRole。

Tools for PowerShell V5

示例 1：此示例从名为“MyRoleName”的角色中删除标签键为“abac”的标签。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMRoleTag -RoleName MyRoleName -TagKey "abac","xyzw"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UntagRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UntagUser** 与 CLI 配合使用

以下代码示例演示如何使用 UntagUser。

CLI

Amazon CLI

删除用户的标签

以下 untag-user 命令从指定用户中移除带有键名称“Department”的任何标签。

```
aws iam untag-user \
    --user-name alice \
    --tag-keys Department
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的标记 IAM 资源。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UntagUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例从名为“joe”的用户中删除标签键为“abac”和“xyzw”的标签。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UntagUser。

Tools for PowerShell V5

示例 1：此示例从名为“joe”的用户中删除标签键为“abac”和“xyzw”的标签。要删除多个标签，请提供以逗号分隔的标签键列表。

```
Remove-IAMUserTag -UserName joe -TagKey "abac","xyzw"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UntagUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateAccessKey** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateAccessKey。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 管理访问密钥

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
##############################################################################
# function iam_update_access_key
#
```

```
# This function can activate or deactivate an IAM access key for the specified
 IAM user.
#
# Parameters:
#       -u user_name  -- The name of the user.
#       -k access_key -- The access key to update.
#       -a            -- Activate the selected access key.
#       -d            -- Deactivate the selected access key.
#
# Example:
#       # To deactivate the selected access key for IAM user Bob
#       iam_update_access_key -u Bob -k AKIAIOSFODNN7EXAMPLE -d
#
# Returns:
#       0 - If successful.
#       1 - If it fails.
###############################################################################
function iam_update_access_key() {
  local user_name access_key status response
  local option OPTARG # Required to use getopts command in a function.
  local activate_flag=false deactivate_flag=false

  # bashsupport disable=BP5008
  function usage() {
    echo "function iam_update_access_key"
    echo "Updates the status of an AWS Identity and Access Management (IAM)
 access key for the specified IAM user"
    echo "  -u user_name    The name of the user."
    echo "  -k access_key   The access key to update."
    echo "  -a              Activate the access key."
    echo "  -d              Deactivate the access key."
    echo ""
  }

  # Retrieve the calling parameters.
    while getopts "u:k:adh" option; do
      case "${option}" in
        u) user_name="${OPTARG}" ;;
        k) access_key="${OPTARG}" ;;
        a) activate_flag=true ;;
        d) deactivate_flag=true ;;
        h)
          usage
          return 0
```

```
          ;;
        \?)
          echo "Invalid parameter"
          usage
          return 1
          ;;
      esac
    done
    export OPTIND=1

  # Validate input parameters
    if [[ -z "$user_name" ]]; then
      errecho "ERROR: You must provide a username with the -u parameter."
      usage
      return 1
    fi

    if [[ -z "$access_key" ]]; then
      errecho "ERROR: You must provide an access key with the -k parameter."
      usage
      return 1
    fi

    # Ensure that only -a or -d is specified
    if [[ "$activate_flag" == true && "$deactivate_flag" == true ]]; then
      errecho "ERROR: You cannot specify both -a (activate) and -d (deactivate)
at the same time."
      usage
      return 1
    fi

    # If neither -a nor -d is provided, return an error
    if [[ "$activate_flag" == false && "$deactivate_flag" == false ]]; then
      errecho "ERROR: You must specify either -a (activate) or -d (deactivate)."
      usage
      return 1
    fi

    # Determine the status based on the flag
    if [[ "$activate_flag" == true ]]; then
      status="Active"
    elif [[ "$deactivate_flag" == true ]]; then
      status="Inactive"
    fi
```

```
    iecho "Parameters:\n"
    iecho "    Username:   $user_name"
    iecho "    Access key: $access_key"
    iecho "    New status: $status"
    iecho ""

    # Update the access key status
    response=$(aws iam update-access-key \
      --user-name "$user_name" \
      --access-key-id "$access_key" \
      --status "$status" 2>&1)

    local error_code=${?}

    if [[ $error_code -ne 0 ]]; then
      aws_cli_error_log $error_code
      errecho "ERROR: AWS reports update-access-key operation failed.\n$response"
      return 1
    fi

    iecho "update-access-key response: $response"
    iecho

    return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateAccessKey。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::updateAccessKey(const Aws::String &userName,
                                  const Aws::String &accessKeyID,
```

```
                                    Aws::IAM::Model::StatusType status,
                                    const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateAccessKeyRequest request;
    request.SetUserName(userName);
    request.SetAccessKeyId(accessKeyID);
    request.SetStatus(status);

    auto outcome = iam.UpdateAccessKey(request);
    if (outcome.IsSuccess()) {
        std::cout << "Successfully updated status of access key "
                  << accessKeyID << " for user " << userName << std::endl;
    }
    else {
        std::cerr << "Error updated status of access key " << accessKeyID <<
                  " for user " << userName << ": " <<
                  outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 UpdateAccessKey。

CLI

Amazon CLI

激活或停用 IAM 用户的访问密钥

以下 `update-access-key` 命令停用名为 Bob 的 IAM 用户的指定访问密钥（访问密钥 ID 和秘密访问密钥）。

```
aws iam update-access-key \
    --access-key-id AKIAIOSFODNN7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

此命令不生成任何输出。

停用密钥意味着它不能用于以编程方式访问 Amazon。但密钥仍然可用且可以重新激活。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的访问密钥。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateAccessKey。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.StatusType;
import software.amazon.awssdk.services.iam.model.UpdateAccessKeyRequest;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateAccessKey {

    private static StatusType statusType;

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <username> <accessId> <status>\s

                Where:
```

```
                    username - The name of the user whose key you want to update.
\s
                    accessId - The access key ID of the secret access key you
 want to update.\s
                    status - The status you want to assign to the secret access
 key.\s
                """;

        if (args.length != 3) {
            System.out.println(usage);
            System.exit(1);
        }

        String username = args[0];
        String accessId = args[1];
        String status = args[2];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateKey(iam, username, accessId, status);
        System.out.println("Done");
        iam.close();
    }

    public static void updateKey(IamClient iam, String username, String accessId,
 String status) {
        try {
            if (status.toLowerCase().equalsIgnoreCase("active")) {
                statusType = StatusType.ACTIVE;
            } else if (status.toLowerCase().equalsIgnoreCase("inactive")) {
                statusType = StatusType.INACTIVE;
            } else {
                statusType = StatusType.UNKNOWN_TO_SDK_VERSION;
            }

            UpdateAccessKeyRequest request = UpdateAccessKeyRequest.builder()
                    .accessKeyId(accessId)
                    .userName(username)
                    .status(statusType)
                    .build();

            iam.updateAccessKey(request);
```

```
            System.out.printf("Successfully updated the status of access key %s
 to" +
                    "status %s for user %s", accessId, status, username);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 UpdateAccessKey。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

更新访问密钥。

```
import {
  UpdateAccessKeyCommand,
  IAMClient,
  StatusType,
} from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} userName
 * @param {string} accessKeyId
 */
export const updateAccessKey = (userName, accessKeyId) => {
```

```
  const command = new UpdateAccessKeyCommand({
    AccessKeyId: accessKeyId,
    Status: StatusType.Inactive,
    UserName: userName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-updating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateAccessKey。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  AccessKeyId: "ACCESS_KEY_ID",
  Status: "Active",
  UserName: "USER_NAME",
};

iam.updateAccessKey(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
```

```
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-access-keys.html#iam-examples-managing-access-keys-updating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateAccessKey。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **Bob** 的 IAM 用户的访问密钥 **AKIAIOSFODNN7EXAMPLE** 状态更改为 **Inactive**。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateAccessKey。

Tools for PowerShell V5

示例 1：此示例将名为 **Bob** 的 IAM 用户的访问密钥 **AKIAIOSFODNN7EXAMPLE** 状态更改为 **Inactive**。

```
Update-IAMAccessKey -UserName Bob -AccessKeyId AKIAIOSFODNN7EXAMPLE -Status
 Inactive
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 UpdateAccessKey。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 UpdateAccessKey。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
TRY.
    lo_iam->updateaccesskey(
      iv_accesskeyid = iv_access_key_id
      iv_status = iv_status
      iv_username = iv_user_name ).
    MESSAGE 'Access key updated successfully.' TYPE 'I'.
  CATCH /aws1/cx_iamnosuchentityex.
    MESSAGE 'Access key or user does not exist.' TYPE 'E'.
ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 UpdateAccessKey。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateAccountPasswordPolicy** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateAccountPasswordPolicy。

CLI

Amazon CLI

设置或更改当前账户密码策略

以下 update-account-password-policy 命令将密码策略设置为要求长度最少为八个字符，并要求在密码中包含一个或多个数字。

```
aws iam update-account-password-policy \
```

```
    --minimum-password-length 8 \
    --require-numbers
```

此命令不生成任何输出。

对账户密码策略的更改会影响为账户中的 IAM 用户创建的所有新密码。密码策略更改不会影响现有的密码。

有关更多信息，请参阅《Amazon IAM 用户指南》中的为 IAM 用户设置账户密码策略。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateAccountPasswordPolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例使用指定设置更新账户的密码策略。请注意，命令中未包含的任何参数都不会保持不变。相反，这些参数将重置为默认值。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateAccountPasswordPolicy。

Tools for PowerShell V5

示例 1：此示例使用指定设置更新账户的密码策略。请注意，命令中未包含的任何参数都不会保持不变。相反，这些参数将重置为默认值。

```
Update-IAMAccountPasswordPolicy -AllowUsersToChangePasswords $true -HardExpiry
 $false -MaxPasswordAge 90 -MinimumPasswordLength 8 -PasswordReusePrevention 20
 -RequireLowercaseCharacters $true -RequireNumbers $true -RequireSymbols $true -
RequireUppercaseCharacters $true
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateAccountPasswordPolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateAssumeRolePolicy** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateAssumeRolePolicy。

CLI

Amazon CLI

更新 IAM 角色的信任策略

以下 update-assume-role-policy 命令更新名为 Test-Role 的角色的信任策略。

```
aws iam update-assume-role-policy \
    --role-name Test-Role \
    --policy-document file://Test-Role-Trust-Policy.json
```

此命令不生成任何输出。

信任策略在 Test-Role-Trust-Policy.json 文件中定义为 JSON 文档。（文件名和扩展名没有意义。）信任策略必须指定主体。

要更新角色的权限策略，请使用 put-role-policy 命令。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateAssumeRolePolicy。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例使用新的信任策略更新名为 **ClientRole** 的 IAM 角色，其内容来自文件 **ClientRolePolicy.json**。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateAssumeRolePolicy。

Tools for PowerShell V5

示例 1：此示例使用新的信任策略更新名为 **ClientRole** 的 IAM 角色，其内容来自文件 **ClientRolePolicy.json**。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。

```
Update-IAMAssumeRolePolicy -RoleName ClientRole -PolicyDocument (Get-Content -raw
 ClientRolePolicy.json)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateAssumeRolePolicy。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateGroup** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateGroup。

CLI

Amazon CLI

重命名 IAM 组

以下 update-group 命令可将 IAM 组 Test 的名称更改为 Test-1。

```
aws iam update-group \
    --group-name Test \
    --new-group-name Test-1
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的重命名 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateGroup。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 IAM 组 **Testers** 重命名为 **AppTesters**。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

示例 2：此示例将 IAM 组 **AppTesters** 的路径更改为 **/Org1/Org2/**。这会将该组的 ARN 更改为 **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateGroup。

Tools for PowerShell V5

示例 1：此示例将 IAM 组 **Testers** 重命名为 **AppTesters**。

```
Update-IAMGroup -GroupName Testers -NewGroupName AppTesters
```

示例 2：此示例将 IAM 组 **AppTesters** 的路径更改为 **/Org1/Org2/**。这会将该组的 ARN 更改为 **arn:aws:iam::123456789012:group/Org1/Org2/AppTesters**。

```
Update-IAMGroup -GroupName AppTesters -NewPath /Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateGroup。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateLoginProfile** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateLoginProfile。

CLI

Amazon CLI

更新 IAM 用户的密码

以下 update-login-profile 命令为名为 Bob 的 IAM 用户创建新密码。

```
aws iam update-login-profile \
    --user-name Bob \
```

```
    --password <password>
```

此命令不生成任何输出。

要为账户设置密码策略，请使用 update-account-password-policy 命令。如果新密码违反了账户密码策略，则该命令将返回 PasswordPolicyViolation 错误。

如果账户密码策略允许，则 IAM 用户可以使用 change-password 命令更改自己的密码。

将密码保存在安全位置。如果密码丢失，则将无法恢复，必须使用 create-login-profile 命令创建新密码。

有关更多信息，请参阅《Amazon IAM 用户指南》中的管理 IAM 用户的密码。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateLoginProfile。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例为 IAM 用户 **Bob** 设置了一个新的临时密码，并要求该用户在下次登录时更改密码。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateLoginProfile。

Tools for PowerShell V5

示例 1：此示例为 IAM 用户 **Bob** 设置了一个新的临时密码，并要求该用户在下次登录时更改密码。

```
Update-IAMLoginProfile -UserName Bob -Password "P@ssw0rd1234" -
PasswordResetRequired $true
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateLoginProfile。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateOpenIdConnectProviderThumbprint** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateOpenIdConnectProviderThumbprint。

CLI

Amazon CLI

将现有服务器证书指纹列表替换为新列表

此示例更新了其 ARN 为 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com 的 OIDC 提供者的证书指纹列表以使用新指纹。

```
aws iam update-open-id-connect-provider-thumbprint \
    --open-id-connect-provider-arn arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com \
    --thumbprint-list 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

如果您的 OIDC 提供者的发现和 JWKS 端点使用不同的证书，请在 --thumbprint-list 值中包含两个端点的指纹。

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 OpenID Connect（OIDC）身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateOpenIdConnectProviderThumbprint。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例更新了其 ARN 为 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 OIDC 提供者的证书指纹列表以使用新指纹。当与提供者关联的证书发生变化时，OIDC 提供者将共享新值。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateOpenIdConnectProviderThumbprint。

Tools for PowerShell V5

示例 1：此示例更新了其 ARN 为 **arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com** 的 OIDC 提供者的证书指纹列表以使用新指纹。当与提供者关联的证书发生变化时，OIDC 提供者将共享新值。

```
Update-IAMOpenIDConnectProviderThumbprint -OpenIDConnectProviderArn
 arn:aws:iam::123456789012:oidc-provider/example.oidcprovider.com -ThumbprintList
 7359755EXAMPLEabc3060bce3EXAMPLEec4542a3
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateOpenIdConnectProviderThumbprint。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateRole** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateRole。

CLI

Amazon CLI

更改 IAM 角色的描述或会话持续时间

以下 update-role 命令将 IAM 角色 production-role 的描述更改为 Main production role，并将最长会话持续时间设置为 12 小时。

```
aws iam update-role \
    --role-name production-role \
    --description 'Main production role' \
    --max-session-duration 43200
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例更新了角色描述和可以请求的角色会话的最长会话持续时间值（以秒为单位）。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateRole。

Tools for PowerShell V5

示例 1：此示例更新了角色描述和可以请求的角色会话的最长会话持续时间值（以秒为单位）。

```
Update-IAMRole -RoleName MyRoleName -Description "My testing role" -
MaxSessionDuration 43200
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateRoleDescription** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateRoleDescription。

CLI

Amazon CLI

更改 IAM 角色的描述

以下 update-role 命令可将 IAM 角色 production-role 的描述更改为 Main production role。

```
aws iam update-role-description \
    --role-name production-role \
```

```
    --description 'Main production role'
```

输出：

```
{
    "Role": {
        "Path": "/",
        "RoleName": "production-role",
        "RoleId": "AROA1234567890EXAMPLE",
        "Arn": "arn:aws:iam::123456789012:role/production-role",
        "CreateDate": "2017-12-06T17:16:37+00:00",
        "AssumeRolePolicyDocument": {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {
                        "AWS": "arn:aws:iam::123456789012:root"
                    },
                    "Action": "sts:AssumeRole",
                    "Condition": {}
                }
            ]
        },
        "Description": "Main production role"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的修改角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateRoleDescription。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例更新了您账户中 IAM 角色的描述。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateRoleDescription。

Tools for PowerShell V5

示例 1：此示例更新了您账户中 IAM 角色的描述。

```
Update-IAMRoleDescription -RoleName MyRoleName -Description "My testing role"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateRoleDescription。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateSamlProvider** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateSamlProvider。

CLI

Amazon CLI

更新现有 SAML 提供者的元数据文档

此示例使用文件 SAMLMetaData.xml 中的新 SAML 元数据文档更新 ARN 为 arn:aws:iam::123456789012:saml-provider/SAMLADFS 的 IAM 中的 SAML 提供者。

```
aws iam update-saml-provider \
    --saml-metadata-document file://SAMLMetaData.xml \
    --saml-provider-arn arn:aws:iam::123456789012:saml-provider/SAMLADFS
```

输出：

```
{
    "SAMLProviderArn": "arn:aws:iam::123456789012:saml-provider/SAMLADFS"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的创建 IAM SAML 身份提供者。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateSamlProvider。

PowerShell

适用于 PowerShell V4 的工具

示例 1 ：此示例使用文件 **SAMLMetaData.xml** 中的新 SAML 元数据文档更新 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFS** 的 IAM 中的 SAML 提供者。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateSamlProvider。

Tools for PowerShell V5

示例 1 ：此示例使用文件 **SAMLMetaData.xml** 中的新 SAML 元数据文档更新 ARN 为 **arn:aws:iam::123456789012:saml-provider/SAMLADFS** 的 IAM 中的 SAML 提供者。请注意，必须使用 **-Raw** 开关参数才能成功处理 JSON 文件的内容。

```
Update-IAMSAMLProvider -SAMLProviderArn arn:aws:iam::123456789012:saml-provider/
SAMLADFS -SAMLMetadataDocument (Get-Content -Raw SAMLMetaData.xml)
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateSamlProvider。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateServerCertificate** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateServerCertificate。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::updateServerCertificate(const Aws::String
 &currentCertificateName,
                                           const Aws::String &newCertificateName,
                                           const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::IAM::IAMClient iam(clientConfig);
    Aws::IAM::Model::UpdateServerCertificateRequest request;
    request.SetServerCertificateName(currentCertificateName);
    request.SetNewServerCertificateName(newCertificateName);

    auto outcome = iam.UpdateServerCertificate(request);
    bool result = true;
    if (outcome.IsSuccess()) {
        std::cout << "Server certificate " << currentCertificateName
                  << " successfully renamed as " << newCertificateName
                  << std::endl;
    }
    else {
        if (outcome.GetError().GetErrorType() !=
 Aws::IAM::IAMErrors::NO_SUCH_ENTITY) {
            std::cerr << "Error changing name of server certificate " <<
                      currentCertificateName << " to " << newCertificateName <<
 ":" <<
                      outcome.GetError().GetMessage() << std::endl;
            result = false;
        }
        else {
            std::cout << "Certificate '" << currentCertificateName
                      << "' not found." << std::endl;
        }
    }

    return result;
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 UpdateServerCertificate。

CLI

Amazon CLI

更改 Amazon 账户中服务器证书的路径或名称

以下 update-server-certificate 命令可将证书名称从 myServerCertificate 更改为 myUpdatedServerCertificate，还会将路径更改为 /cloudfront/，以便 Amazon CloudFront 服务可以访问它。此命令不生成任何输出。运行 list-server-certificates 命令即可查看更新结果。

```
aws-iam update-server-certificate \
    --server-certificate-name myServerCertificate \
    --new-server-certificate-name myUpdatedServerCertificate \
    --new-path /cloudfront/
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的在 IAM 中管理服务器证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateServerCertificate。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

更新服务器证书。

```
import { UpdateServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentName
```

```
 * @param {string} newName
 */
export const updateServerCertificate = (currentName, newName) => {
  const command = new UpdateServerCertificateCommand({
    ServerCertificateName: currentName,
    NewServerCertificateName: newName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-updating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateServerCertificate。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  ServerCertificateName: "CERTIFICATE_NAME",
  NewServerCertificateName: "NEW_CERTIFICATE_NAME",
};

iam.updateServerCertificate(params, function (err, data) {
  if (err) {
    console.log("Error", err);
```

```
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-server-certificates.html#iam-examples-server-certificates-updating。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateServerCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将名为 **MyServerCertificate** 的证书重命名为 **MyRenamedServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

示例 2：此示例将名为 **MyServerCertificate** 的证书移动到路径 /Org1/Org2/。这会将该资源的 ARN 更改为 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateServerCertificate。

Tools for PowerShell V5

示例 1：此示例将名为 **MyServerCertificate** 的证书重命名为 **MyRenamedServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -
NewServerCertificateName MyRenamedServerCertificate
```

示例 2：此示例将名为 **MyServerCertificate** 的证书移动到路径 /Org1/Org2/。这会将该资源的 ARN 更改为 **arn:aws:iam::123456789012:server-certificate/Org1/Org2/MyServerCertificate**。

```
Update-IAMServerCertificate -ServerCertificateName MyServerCertificate -NewPath /
Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 UpdateServerCertificate。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

列出、更新和删除服务器证书。

```
class ServerCertificateManager
  def initialize(iam_client, logger: Logger.new($stdout))
    @iam_client = iam_client
    @logger = logger
    @logger.progname = 'ServerCertificateManager'
  end

  # Creates a new server certificate.
  # @param name [String] the name of the server certificate
  # @param certificate_body [String] the contents of the certificate
  # @param private_key [String] the private key contents
  # @return [Boolean] returns true if the certificate was successfully created
  def create_server_certificate(name, certificate_body, private_key)
    @iam_client.upload_server_certificate({
                                            server_certificate_name: name,
                                            certificate_body: certificate_body,
                                            private_key: private_key
                                          })
    true
```

```
  rescue Aws::IAM::Errors::ServiceError => e
    puts "Failed to create server certificate: #{e.message}"
    false
  end

  # Lists available server certificate names.
  def list_server_certificate_names
    response = @iam_client.list_server_certificates

    if response.server_certificate_metadata_list.empty?
      @logger.info('No server certificates found.')
      return
    end

    response.server_certificate_metadata_list.each do |certificate_metadata|
      @logger.info("Certificate Name:
#{certificate_metadata.server_certificate_name}")
    end
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error listing server certificates: #{e.message}")
  end

  # Updates the name of a server certificate.
  def update_server_certificate_name(current_name, new_name)
    @iam_client.update_server_certificate(
      server_certificate_name: current_name,
      new_server_certificate_name: new_name
    )
    @logger.info("Server certificate name updated from '#{current_name}' to
'#{new_name}'.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error updating server certificate name: #{e.message}")
    false
  end

  # Deletes a server certificate.
  def delete_server_certificate(name)
    @iam_client.delete_server_certificate(server_certificate_name: name)
    @logger.info("Server certificate '#{name}' deleted.")
    true
  rescue Aws::IAM::Errors::ServiceError => e
    @logger.error("Error deleting server certificate: #{e.message}")
    false
```

```
  end
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 UpdateServerCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateSigningCertificate** 与 CLI 配合使用

以下代码示例演示如何使用 UpdateSigningCertificate。

CLI

Amazon CLI

激活或停用 IAM 用户的签名证书

以下 update-signing-certificate 命令停用名为 Bob 的 IAM 用户的指定签名证书。

```
aws iam update-signing-certificate \
    --certificate-id TA7SMP42TDN5Z26OBPJE7EXAMPLE \
    --status Inactive \
    --user-name Bob
```

要获取签名证书的 ID，请使用 list-signing-certificates 命令。

有关更多信息，请参阅《Amazon EC2 用户指南》中的管理签名证书。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateSigningCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例更新了与名为 **Bob** 且其证书 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的 IAM 用户关联的证书，以将其标记为非活动状态。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UpdateSigningCertificate。

Tools for PowerShell V5

示例 1：此示例更新了与名为 **Bob** 且其证书 ID 为 **Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU** 的 IAM 用户关联的证书，以将其标记为非活动状态。

```
Update-IAMSigningCertificate -CertificateId Y3EK7RMEXAMPLESV33FCREXAMPLEMJLU -
UserName Bob -Status Inactive
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UpdateSigningCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateUser** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 UpdateUser。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 创建只读和读写用户

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::IAM::updateUser(const Aws::String &currentUserName,
                             const Aws::String &newUserName,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
```

```
    Aws::IAM::IAMClient iam(clientConfig);

    Aws::IAM::Model::UpdateUserRequest request;
    request.SetUserName(currentUserName);
    request.SetNewUserName(newUserName);

    auto outcome = iam.UpdateUser(request);
    if (outcome.IsSuccess()) {
        std::cout << "IAM user " << currentUserName <<
                  " successfully updated with new user name " << newUserName <<
                  std::endl;
    }
    else {
        std::cerr << "Error updating user name for IAM user " << currentUserName
 <<
                  ":" << outcome.GetError().GetMessage() << std::endl;
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 UpdateUser。

CLI

Amazon CLI

更改 IAM 用户的名称

以下 update-user 命令将 IAM 用户 Bob 的名称更改为 Robert。

```
aws iam update-user \
    --user-name Bob \
    --new-user-name Robert
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon IAM 用户指南》中的重命名 IAM 用户组。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UpdateUser。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.IamException;
import software.amazon.awssdk.services.iam.model.UpdateUserRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class UpdateUser {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <curName> <newName>\s

                Where:
                    curName - The current user name.\s
                    newName - An updated user name.\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String curName = args[0];
```

```
        String newName = args[1];
        Region region = Region.AWS_GLOBAL;
        IamClient iam = IamClient.builder()
                .region(region)
                .build();

        updateIAMUser(iam, curName, newName);
        System.out.println("Done");
        iam.close();
    }

    public static void updateIAMUser(IamClient iam, String curName, String
 newName) {
        try {
            UpdateUserRequest request = UpdateUserRequest.builder()
                    .userName(curName)
                    .newUserName(newName)
                    .build();

            iam.updateUser(request);
            System.out.printf("Successfully updated user to username %s",
 newName);

        } catch (IamException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 UpdateUser。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

更新用户。

```
import { UpdateUserCommand, IAMClient } from "@aws-sdk/client-iam";

const client = new IAMClient({});

/**
 *
 * @param {string} currentUserName
 * @param {string} newUserName
 */
export const updateUser = (currentUserName, newUserName) => {
  const command = new UpdateUserCommand({
    UserName: currentUserName,
    NewUserName: newUserName,
  });

  return client.send(command);
};
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v3/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-updating-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateUser。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
var AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

// Create the IAM service object
```

```
var iam = new AWS.IAM({ apiVersion: "2010-05-08" });

var params = {
  UserName: process.argv[2],
  NewUserName: process.argv[3],
};

iam.updateUser(params, function (err, data) {
  if (err) {
    console.log("Error", err);
  } else {
    console.log("Success", data);
  }
});
```

- 有关更多信息，请参阅《适用于 JavaScript 的 Amazon SDK 开发人员指南》https://docs.amazonaws.cn/sdk-for-javascript/v2/developer-guide/iam-examples-managing-users.html#iam-examples-managing-users-updating-users。
- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UpdateUser。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
suspend fun updateIAMUser(
    curName: String?,
    newName: String?,
) {
    val request =
        UpdateUserRequest {
            userName = curName
            newUserName = newName
```

```
        }

    IamClient.fromEnvironment { region = "AWS_GLOBAL" }.use { iamClient ->
        iamClient.updateUser(request)
        println("Successfully updated user to $newName")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API Reference》中的 UpdateUser。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将 IAM 用户 **Bob** 重命名为 **Robert**。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

示例 2：此示例将 IAM 用户 **Bob** 的路径更改为 **/Org1/Org2/**，这实际上将用户的 ARN 更改为 **arn:aws:iam::123456789012:user/Org1/Org2/bob**。

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell CCmdlet Reference (V4)》中的 UpdateUser。

Tools for PowerShell V5

示例 1：此示例将 IAM 用户 **Bob** 重命名为 **Robert**。

```
Update-IAMUser -UserName Bob -NewUserName Robert
```

示例 2：此示例将 IAM 用户 **Bob** 的路径更改为 **/Org1/Org2/**，这实际上将用户的 ARN 更改为 **arn:aws:iam::123456789012:user/Org1/Org2/bob**。

```
Update-IAMUser -UserName Bob -NewPath /Org1/Org2/
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 UpdateUser。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def update_user(user_name, new_user_name):
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的 UpdateUser。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
# Updates an IAM user's name
#
# @param current_name [String] The current name of the user
# @param new_name [String] The new name of the user
def update_user_name(current_name, new_name)
  @iam_client.update_user(user_name: current_name, new_user_name: new_name)
  true
rescue StandardError => e
  @logger.error("Error updating user name from '#{current_name}' to
'#{new_name}': #{e.message}")
  false
end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 UpdateUser。

SAP ABAP

适用于 SAP ABAP 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
  TRY.
      lo_iam->updateuser(
        iv_username = iv_user_name
```

```
          iv_newusername = iv_new_user_name ).
        MESSAGE 'User updated successfully.' TYPE 'I'.
      CATCH /aws1/cx_iamnosuchentityex.
        MESSAGE 'User does not exist.' TYPE 'E'.
      CATCH /aws1/cx_iamentityalrdyexex.
        MESSAGE 'New user name already exists.' TYPE 'E'.
    ENDTRY.
```

- 有关 API 详细信息，请参阅《Amazon SDK for SAP ABAP API Reference》中的 UpdateUser。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadServerCertificate** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 UploadServerCertificate。

CLI

Amazon CLI

将服务器证书上传到 Amazon 账户

使用以下 upload-server-certificate 命令可将服务器证书上传到 Amazon 账户。在此示例中，证书位于 public_key_cert_file.pem 文件中，关联的私有密钥位于 my_private_key.pem 文件中，而证书颁发机构（CA）提供的证书链位于 my_certificate_chain_file.pem 文件中。文件上传完成后，就位于 myServerCertificate 名称下。以 file:// 开头的参数让命令读取文件的内容，将其用作参数值，而不是文件名本身。

```
aws iam upload-server-certificate \
    --server-certificate-name myServerCertificate \
    --certificate-body file://public_key_cert_file.pem \
    --private-key file://my_private_key.pem \
    --certificate-chain file://my_certificate_chain_file.pem
```

输出：

```
{
```

```
    "ServerCertificateMetadata": {
        "Path": "/",
        "ServerCertificateName": "myServerCertificate",
        "ServerCertificateId": "ASCAEXAMPLE123EXAMPLE",
        "Arn": "arn:aws:iam::1234567989012:server-certificate/
myServerCertificate",
        "UploadDate": "2019-04-22T21:13:44+00:00",
        "Expiration": "2019-10-15T22:23:16+00:00"
    }
}
```

有关更多信息，请参阅《使用 IAM》指南中的“创建、上传和删除服务器证书”。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UploadServerCertificate。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import { UploadServerCertificateCommand, IAMClient } from "@aws-sdk/client-iam";
import { readFileSync } from "node:fs";
import { dirnameFromMetaUrl } from "@aws-doc-sdk-examples/lib/utils/util-fs.js";
import * as path from "node:path";

const client = new IAMClient({});

const certMessage = `Generate a certificate and key with the following command,
 or the equivalent for your system.

openssl req -x509 -newkey rsa:4096 -sha256 -days 3650 -nodes \
-keyout example.key -out example.crt -subj "/CN=example.com" \
-addext "subjectAltName=DNS:example.com,DNS:www.example.net,IP:10.0.0.1"
`;

const getCertAndKey = () => {
  try {
```

```
    const cert = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.crt"),
    );
    const key = readFileSync(
      path.join(dirnameFromMetaUrl(import.meta.url), "./example.key"),
    );
    return { cert, key };
  } catch (err) {
    if (err.code === "ENOENT") {
      throw new Error(
        `Certificate and/or private key not found. ${certMessage}`,
      );
    }

    throw err;
  }
};

/**
 *
 * @param {string} certificateName
 */
export const uploadServerCertificate = (certificateName) => {
  const { cert, key } = getCertAndKey();
  const command = new UploadServerCertificateCommand({
    ServerCertificateName: certificateName,
    CertificateBody: cert.toString(),
    PrivateKey: key.toString(),
  });

  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 UploadServerCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例将新服务器证书上传到 IAM 账户。包含证书正文、私有密钥和（可选）证书链的文件必须均采用 PEM 编码。请注意，参数需要文件的实际内容，而不是文件名。必须使用 **-Raw** 开关参数才能成功处理文件内容。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UploadServerCertificate。

Tools for PowerShell V5

示例 1：此示例将新服务器证书上传到 IAM 账户。包含证书正文、私有密钥和（可选）证书链的文件必须均采用 PEM 编码。请注意，参数需要文件的实际内容，而不是文件名。必须使用 **-Raw** 开关参数才能成功处理文件内容。

```
Publish-IAMServerCertificate -ServerCertificateName MyTestCert -CertificateBody
 (Get-Content -Raw server.crt) -PrivateKey (Get-Content -Raw server.key)
```

输出：

```
Arn                   : arn:aws:iam::123456789012:server-certificate/MyTestCert
Expiration            : 1/14/2018 9:52:36 AM
Path                  : /
ServerCertificateId   : ASCAJIEXAMPLE7J7HQZYW
ServerCertificateName : MyTestCert
UploadDate            : 4/21/2015 11:14:16 AM
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UploadServerCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UploadSigningCertificate** 与 CLI 配合使用

以下代码示例演示如何使用 UploadSigningCertificate。

CLI

Amazon CLI

上传 IAM 用户的签名证书

以下 upload-signing-certificate 命令上传名为 Bob 的 IAM 用户的签名证书。

```
aws iam upload-signing-certificate \
    --user-name Bob \
    --certificate-body file://certificate.pem
```

输出：

```
{
    "Certificate": {
        "UserName": "Bob",
        "Status": "Active",
        "CertificateBody": "-----BEGIN CERTIFICATE-----<certificate-body>-----END
 CERTIFICATE-----",
        "CertificateId": "TA7SMP42TDN5Z26OBPJE7EXAMPLE",
        "UploadDate": "2013-06-06T21:40:08.121Z"
    }
}
```

证书位于名为 certificate.pem 的文件中，采用 PEM 格式。

有关更多信息，请参阅《使用 IAM》指南中的“创建和上传用户签名证书”。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 UploadSigningCertificate。

PowerShell

适用于 PowerShell V4 的工具

示例 1：此示例上传新的 X.509 签名证书并将其与名为 **Bob** 的 IAM 用户关联。包含证书正文的文件采用 PEM 编码。**CertificateBody** 参数需要证书文件的实际内容而不是文件名。必须使用 **-Raw** 开关参数才能成功处理该文件。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                   YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                   21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                   rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                   nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb

 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                   NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                   -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
```

```
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 UploadSigningCertificate。

Tools for PowerShell V5

示例 1：此示例上传新的 X.509 签名证书并将其与名为 **Bob** 的 IAM 用户关联。包含证书正文的文件采用 PEM 编码。**CertificateBody** 参数需要证书文件的实际内容而不是文件名。必须使用 **-Raw** 开关参数才能成功处理该文件。

```
Publish-IAMSigningCertificate -UserName Bob -CertificateBody (Get-Content -Raw
 SampleSigningCert.pem)
```

输出：

```
CertificateBody : -----BEGIN CERTIFICATE-----

 MIICiTCCAfICCQD6m7oRw0uXOjANBgkqhkiG9w0BAQUFADCBiDELMAkGA1UEBhMC

 VVMxCzAJBgNVBAgTAldBMRAwDgYDVQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6

 b24xFDASBgNVBAsTC0lBTSBDb25zb2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAd

 BgkqhkiG9w0BCQEWEG5vb25lQGFtYXpvbi5jb20wHhcNMTEwNDI1MjA0NTIxWhcN

 MTIwNDI0MjA0NTIxWjCBiDELMAkGA1UEBhMCVVMxCzAJBgNVBAgTAldBMRAwDgYD

 VQQHEwdTZWF0dGxlMQ8wDQYDVQQKEwZBbWF6b24xFDASBgNVBAsTC0lBTSBDb25z

 b2xlMRIwEAYDVQQDEwlUZXN0Q2lsYWMxHzAdBgkqhkiG9w0BCQEWEG5vb25lQGFt
                    YXpvbi5jb20wgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAMaK0dn
+a4GmWIWJ
                    21uUSfwfEvySWtC2XADZ4nB+BLYgVIk60CpiwsZ3G93vUEIO3IyNoH/
f0wYK8m9T
                    rDHudUZg3qX4waLG5M43q7Wgc/
MbQITxOUSQv7c7ugFFDzQGBzZswY6786m86gpE

 Ibb3OhjZnzcvQAaRHhdlQWIMm2nrAgMBAAEwDQYJKoZIhvcNAQEFBQADgYEAtCu4
                    nUhVVxYUntneD9+h8Mg9q6q
+auNKyExzyLwaxlAoo7TJHidbtS4J5iNmZgXL0Fkb
```

```
 FFBjvSfpJIlJ00zbhNYS5f6GuoEDmFJl0ZxBHjJnyp378OD8uTs7fLvjx79LjSTb
                  NYiytVbZPQUQ5Yaxu2jXnimvw3rrszlaEXAMPLE=
                  -----END CERTIFICATE-----
CertificateId   : Y3EK7RMEXAMPLESV33FCEXAMPLEHMJLU
Status          : Active
UploadDate      : 4/20/2015 1:26:01 PM
UserName        : Bob
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 UploadSigningCertificate。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包的 IAM 场景

以下代码示例显示如何通过 Amazon 开发工具包实施 IAM 中的常见场景。这些场景向您展示了如何通过调用 IAM 中的多个函数或与其他 Amazon Web Services 服务 结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用 Amazon 开发工具包构建和管理弹性服务
- 配置 Amazon ECS Service Connect
- 使用 Lambda 代理集成创建 REST API
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 使用 Amazon 开发工具包创建只读和读写 IAM 用户
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 EC2 启动类型的 Amazon ECS 服务
- 创建 Amazon Managed Grafana 工作区
- 创建第一个 Lambda 函数
- 通过 CLI 开始使用 Redshift Serverless
- IoT Device Defender 入门
- 开始使用 Amazon EKS

- Amazon MSK 入门
- Amazon Redshift 预置集群入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- Step Functions 入门
- 使用 Amazon 开发工具包管理 IAM 访问密钥
- 使用 Amazon 开发工具包管理 IAM policy
- 使用 Amazon 开发工具包管理 IAM 角色
- 使用 Amazon 开发工具包管理您的 IAM 账户
- 将硬编码的密钥移动到 Secrets Manager
- 允许 Amazon Compute Optimizer 自动化功能应用建议的操作
- 在整个组织中启用自动化功能的策略
- 启用账户自动化功能的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能完全访问权限的策略
- 向组织管理账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 向独立 Amazon 账户授予对 Compute Optimizer 自动化功能只读访问权限的策略
- 授予计算优化自动化功能的服务相关角色权限的策略
- 使用 Amazon SDK 回滚 IAM policy 版本
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 使用 Amazon Command Line Interface v2 为 DynamoDB 设置基于属性的访问控制
- 设置 Systems Manager
- 在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数
- 使用 Amazon Command Line Interface v2 处理 DynamoDB Streams 和生存时间
- 通过 Amazon SDK 使用 IAM Policy Builder API

使用 Amazon 开发工具包构建和管理弹性服务

以下代码示例展示如何创建可返回书籍、电影和歌曲推荐的负载均衡的 Web 服务。该示例演示服务如何响应故障，以及如何重组服务以提高故障发生时的弹性。

- 使用 Amazon EC2 Auto Scaling 组根据启动模板创建 Amazon Elastic Compute Cloud（Amazon EC2）实例，并将实例数量保持在指定范围内。
- 使用弹性负载均衡处理和分发 HTTP 请求。
- 监控自动扩缩组中实例的运行状况，并仅将请求转发到运行状况良好的实例。
- 在每个 EC2 实例上运行 Python Web 服务器以处理 HTTP 请求。Web 服务器以建议和运行状况检查作为响应。
- 使用 Amazon DynamoDB 表模拟推荐服务。
- 通过更新 Amazon Systems Manager 参数控制 Web 服务器对请求和运行状况检查的响应。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
    static async Task Main(string[] args)
    {
        _configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally, load local settings.
            .Build();


        // Set up dependency injection for the AWS services.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
LogLevel.Trace))
```

```
            .ConfigureServices((_, services) =>
                services.AddAWSService<IAmazonIdentityManagementService>()
                    .AddAWSService<IAmazonDynamoDB>()
                    .AddAWSService<IAmazonElasticLoadBalancingV2>()
                    .AddAWSService<IAmazonSimpleSystemsManagement>()
                    .AddAWSService<IAmazonAutoScaling>()
                    .AddAWSService<IAmazonEC2>()
                    .AddTransient<AutoScalerWrapper>()
                    .AddTransient<ElasticLoadBalancerWrapper>()
                    .AddTransient<SmParameterWrapper>()
                    .AddTransient<Recommendations>()
                    .AddSingleton<IConfiguration>(_configuration)
            )
            .Build();

        ServicesSetup(host);
        ResourcesSetup();

        try
        {
            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Welcome to the Resilient Architecture Example
Scenario.");
            Console.WriteLine(new string('-', 80));
            await Deploy(true);

            Console.WriteLine("Now let's begin the scenario.");
            Console.WriteLine(new string('-', 80));
            await Demo(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Finally, let's clean up our resources.");
            Console.WriteLine(new string('-', 80));

            await DestroyResources(true);

            Console.WriteLine(new string('-', 80));
            Console.WriteLine("Resilient Architecture Example Scenario is
complete.");
            Console.WriteLine(new string('-', 80));
        }
        catch (Exception ex)
        {
            Console.WriteLine(new string('-', 80));
```

```
            Console.WriteLine($"There was a problem running the scenario:
{ex.Message}");
            await DestroyResources(true);
            Console.WriteLine(new string('-', 80));
        }
    }

    /// <summary>
    /// Setup any common resources, also used for integration testing.
    /// </summary>
    public static void ResourcesSetup()
    {
        _httpClient = new HttpClient();
    }

    /// <summary>
    /// Populate the services for use within the console application.
    /// </summary>
    /// <param name="host">The services host.</param>
    private static void ServicesSetup(IHost host)
    {
        _elasticLoadBalancerWrapper =
host.Services.GetRequiredService<ElasticLoadBalancerWrapper>();
        _iamClient =
host.Services.GetRequiredService<IAmazonIdentityManagementService>();
        _recommendations = host.Services.GetRequiredService<Recommendations>();
        _autoScalerWrapper =
host.Services.GetRequiredService<AutoScalerWrapper>();
        _smParameterWrapper =
host.Services.GetRequiredService<SmParameterWrapper>();
    }

    /// <summary>
    /// Deploy necessary resources for the scenario.
    /// </summary>
    /// <param name="interactive">True to run as interactive.</param>
    /// <returns>True if successful.</returns>
    public static async Task<bool> Deploy(bool interactive)
    {
        var protocol = "HTTP";
        var port = 80;
        var sshPort = 22;

        Console.WriteLine(
```

```
            "\nFor this demo, we'll use the AWS SDK for .NET to create several
AWS resources\n" +
            "to set up a load-balanced web service endpoint and explore some ways
to make it resilient\n" +
            "against various kinds of failures.\n\n" +
            "Some of the resources create by this demo are:\n");

        Console.WriteLine(
            "\t* A DynamoDB table that the web service depends on to provide
book, movie, and song recommendations.");
        Console.WriteLine(
            "\t* An EC2 launch template that defines EC2 instances that each
contain a Python web server.");
        Console.WriteLine(
            "\t* An EC2 Auto Scaling group that manages EC2 instances across
several Availability Zones.");
        Console.WriteLine(
            "\t* An Elastic Load Balancing (ELB) load balancer that targets the
Auto Scaling group to distribute requests.");
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to start deploying
resources.");
        if (interactive)
            Console.ReadLine();

        // Create and populate the DynamoDB table.
        var databaseTableName = _configuration["databaseName"];
        var recommendationsPath = Path.Join(_configuration["resourcePath"],
            "recommendations_objects.json");
        Console.WriteLine($"Creating and populating a DynamoDB table named
{databaseTableName}.");
        await _recommendations.CreateDatabaseWithName(databaseTableName);
        await _recommendations.PopulateDatabase(databaseTableName,
recommendationsPath);
        Console.WriteLine(new string('-', 80));

        // Create the EC2 Launch Template.

        Console.WriteLine(
            $"Creating an EC2 launch template that runs
'server_startup_script.sh' when an instance starts.\n"
            + "\nThis script starts a Python web server defined in the
`server.py` script. The web server\n"
```

```
            + "listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.\n"
            + "For demo purposes, this server is run as the root user. In
production, the best practice is to\n"
            + "run a web server, such as Apache, with least-privileged
credentials.");
        Console.WriteLine(
            "\nThe template also defines an IAM policy that each instance uses to
assume a role that grants\n"
            + "permissions to access the DynamoDB recommendation table and
Systems Manager parameters\n"
            + "that control the flow of the demo.");

        var startupScriptPath = Path.Join(_configuration["resourcePath"],
            "server_startup_script.sh");
        var instancePolicyPath = Path.Join(_configuration["resourcePath"],
            "instance_policy.json");
        await _autoScalerWrapper.CreateTemplate(startupScriptPath,
instancePolicyPath);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different\n"
            + "Availability Zone.\n");
        var zones = await _autoScalerWrapper.DescribeAvailabilityZones();
        await _autoScalerWrapper.CreateGroupOfSize(3,
_autoScalerWrapper.GroupName, zones);
        Console.WriteLine(new string('-', 80));

        Console.WriteLine(
            "At this point, you have EC2 instances created. Once each instance
starts, it listens for\n"
            + "HTTP requests. You can see these instances in the console or
continue with the demo.\n");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue.");
        if (interactive)
            Console.ReadLine();

        Console.WriteLine("Creating variables that control the flow of the
demo.");
        await _smParameterWrapper.Reset();
```

```
        Console.WriteLine(
            "\nCreating an Elastic Load Balancing target group and load balancer.
 The target group\n"
            + "defines how the load balancer connects to instances. The load
 balancer provides a\n"
            + "single endpoint where clients connect and dispatches requests to
 instances in the group.");

        var defaultVpc = await _autoScalerWrapper.GetDefaultVpc();
        var subnets = await
 _autoScalerWrapper.GetAllVpcSubnetsForZones(defaultVpc.VpcId, zones);
        var subnetIds = subnets.Select(s => s.SubnetId).ToList();
        var targetGroup = await
 _elasticLoadBalancerWrapper.CreateTargetGroupOnVpc(_elasticLoadBalancerWrapper.TargetGro
 protocol, port, defaultVpc.VpcId);

        await
 _elasticLoadBalancerWrapper.CreateLoadBalancerAndListener(_elasticLoadBalancerWrapper.Lo
 subnetIds, targetGroup);
        await
 _autoScalerWrapper.AttachLoadBalancerToGroup(_autoScalerWrapper.GroupName,
 targetGroup.TargetGroupArn);
        Console.WriteLine("\nVerifying access to the load balancer endpoint...");
        var endPoint = await
 _elasticLoadBalancerWrapper.GetEndpointForLoadBalancerByName(_elasticLoadBalancerWrapper
        var loadBalancerAccess = await
 _elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);

        if (!loadBalancerAccess)
        {
            Console.WriteLine("\nCouldn't connect to the load balancer, verifying
 that the port is open...");

            var ipString = await _httpClient.GetStringAsync("https://
checkip.amazonaws.com");
            ipString = ipString.Trim();

            var defaultSecurityGroup = await
 _autoScalerWrapper.GetDefaultSecurityGroupForVpc(defaultVpc);
            var portIsOpen =
 _autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, port,
 ipString);
```

```
            var sshPortIsOpen =
_autoScalerWrapper.VerifyInboundPortForGroup(defaultSecurityGroup, sshPort,
ipString);

            if (!portIsOpen)
            {
                Console.WriteLine(
                    "\nFor this example to work, the default security group for
your default VPC must\n"
                    + "allows access from this computer. You can either add it
automatically from this\n"
                    + "example or add it yourself using the AWS Management
Console.\n");

                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound traffic from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, port,
ipString);
                }
            }

            if (!sshPortIsOpen)
            {
                if (!interactive || GetYesNoResponse(
                        "Do you want to add a rule to the security group to allow
inbound SSH traffic for debugging from your computer's IP address?"))
                {
                    await
_autoScalerWrapper.OpenInboundPort(defaultSecurityGroup.GroupId, sshPort,
ipString);
                }
            }
            loadBalancerAccess = await
_elasticLoadBalancerWrapper.VerifyLoadBalancerEndpoint(endPoint);
        }

        if (loadBalancerAccess)
        {
            Console.WriteLine("Your load balancer is ready. You can access it by
browsing to:");
            Console.WriteLine($"\thttp://{endPoint}\n");
```

```
        }
        else
        {
            Console.WriteLine(
                "\nCouldn't get a successful response from the load balancer
endpoint. Troubleshoot by\n"
                + "manually verifying that your VPC and security group are
configured correctly and that\n"
                + "you can successfully make a GET request to the load balancer
endpoint:\n");
            Console.WriteLine($"\thttp://{endPoint}\n");
        }
        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Press Enter when you're ready to continue with the
demo.");
        if (interactive)
            Console.ReadLine();
        return true;
    }

    /// <summary>
    /// Demonstrate the steps of the scenario.
    /// </summary>
    /// <param name="interactive">True to run as an interactive scenario.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> Demo(bool interactive)
    {
        var ssmOnlyPolicy = Path.Join(_configuration["resourcePath"],
            "ssm_only_policy.json");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Resetting parameters to starting values for demo.");
        await _smParameterWrapper.Reset();

        Console.WriteLine("\nThis part of the demonstration shows how to toggle
different parts of the system\n" +
                          "to create situations where the web service fails, and
shows how using a resilient\n" +
                          "architecture can keep the web service running in spite
of these failures.");
        Console.WriteLine(new string('-', 88));
        Console.WriteLine("At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.");
        if (interactive)
```

```
            await DemoActionChoices();

        Console.WriteLine($"The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.\n" +
                          $"The table name is contained in a Systems Manager
parameter named '{_smParameterWrapper.TableParameter}'.\n" +
                          $"To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as\n" +
                          "healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Instead of failing when the recommendation service
fails, the web service can return a static response.");
        Console.WriteLine("While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.FailureResponseParameter,
"static");

        Console.WriteLine("\nNow, sending a GET request to the load balancer
endpoint returns a static response.");
        Console.WriteLine("The service still reports as healthy because health
checks are still shallow.");
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("Let's reinstate the recommendation service.\n");
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
_smParameterWrapper.TableName);
        Console.WriteLine(
            "\nLet's also substitute bad credentials for one of the instances in
the target group so that it can't\n" +
            "access the DynamoDB recommendation table.\n"
        );
        await _autoScalerWrapper.CreateInstanceProfileWithName(
```

```
            _autoScalerWrapper.BadCredsPolicyName,
            _autoScalerWrapper.BadCredsRoleName,
            _autoScalerWrapper.BadCredsProfileName,
            ssmOnlyPolicy,
            new List<string> { "AmazonSSMManagedInstanceCore" }
        );
        var instances = await
_autoScalerWrapper.GetInstancesByGroupName(_autoScalerWrapper.GroupName);
        var badInstanceId = instances.First();
        var instanceProfile = await
_autoScalerWrapper.GetInstanceProfile(badInstanceId);
        Console.WriteLine(
            $"Replacing the profile for instance {badInstanceId} with a profile
that contains\n" +
            "bad credentials...\n"
        );
        await _autoScalerWrapper.ReplaceInstanceProfile(
            badInstanceId,
            _autoScalerWrapper.BadCredsProfileName,
            instanceProfile.AssociationId
        );
        Console.WriteLine(
            "Now, sending a GET request to the load balancer endpoint returns
either a recommendation or a static response,\n" +
            "depending on which instance is selected by the load balancer.\n"
        );
        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nLet's implement a deep health check. For this demo,
a deep health check tests whether");
        Console.WriteLine("the web service can access the DynamoDB table that it
depends on for recommendations. Note that");
        Console.WriteLine("the deep health check is only for ELB routing and not
for Auto Scaling instance health.");
        Console.WriteLine("This kind of deep health check is not recommended for
Auto Scaling instance health, because it");
        Console.WriteLine("risks accidental termination of all instances in the
Auto Scaling group when a dependent service fails.");

        Console.WriteLine("\nBy implementing deep health checks, the load
balancer can detect when one of the instances is failing");
        Console.WriteLine("and take that instance out of rotation.");
```

```
        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.HealthCheckParameter,
"deep");

        Console.WriteLine($"\nNow, checking target health indicates that the
instance with bad credentials ({badInstanceId})");
        Console.WriteLine("is unhealthy. Note that it might take a minute or two
for the load balancer to detect the unhealthy");
        Console.WriteLine("instance. Sending a GET request to the load balancer
endpoint always returns a recommendation, because");
        Console.WriteLine("the load balancer takes unhealthy instances out of its
rotation.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nBecause the instances in this demo are controlled by
an auto scaler, the simplest way to fix an unhealthy");
        Console.WriteLine("instance is to terminate it and let the auto scaler
start a new instance to replace it.");

        await _autoScalerWrapper.TryTerminateInstanceById(badInstanceId);

        Console.WriteLine($"\nEven while the instance is terminating and the new
instance is starting, sending a GET");
        Console.WriteLine("request to the web service continues to get a
successful recommendation response because");
        Console.WriteLine("starts and reports as healthy, it is included in the
load balancing rotation.");
        Console.WriteLine("Note that terminating and replacing an instance
typically takes several minutes, during which time you");
        Console.WriteLine("can see the changing health check status until the new
instance is running and healthy.");

        if (interactive)
            await DemoActionChoices();

        Console.WriteLine("\nIf the recommendation service fails now, deep health
checks mean all instances report as unhealthy.");

        await
_smParameterWrapper.PutParameterByName(_smParameterWrapper.TableParameter,
"this-is-not-a-table");
```

```
        Console.WriteLine($"\nWhen all instances are unhealthy, the load balancer
continues to route requests even to");
        Console.WriteLine("unhealthy instances, allowing them to fail open and
return a static response rather than fail");
        Console.WriteLine("closed and report failure to the customer.");

        if (interactive)
            await DemoActionChoices();
        await _smParameterWrapper.Reset();

        Console.WriteLine(new string('-', 80));
        return true;
    }

    /// <summary>
    /// Clean up the resources from the scenario.
    /// </summary>
    /// <param name="interactive">True to ask the user for cleanup.</param>
    /// <returns>Async task.</returns>
    public static async Task<bool> DestroyResources(bool interactive)
    {
        Console.WriteLine(new string('-', 80));
        Console.WriteLine(
            "To keep things tidy and to avoid unwanted charges on your account,
we can clean up all AWS resources\n" +
            "that were created for this demo."
        );

        if (!interactive || GetYesNoResponse("Do you want to clean up all demo
resources? (y/n) "))
        {
            await
_elasticLoadBalancerWrapper.DeleteLoadBalancerByName(_elasticLoadBalancerWrapper.LoadBal
            await
_elasticLoadBalancerWrapper.DeleteTargetGroupByName(_elasticLoadBalancerWrapper.TargetGr
            await
_autoScalerWrapper.TerminateAndDeleteAutoScalingGroupWithName(_autoScalerWrapper.GroupNa
            await
_autoScalerWrapper.DeleteKeyPairByName(_autoScalerWrapper.KeyPairName);
            await
_autoScalerWrapper.DeleteTemplateByName(_autoScalerWrapper.LaunchTemplateName);
            await _autoScalerWrapper.DeleteInstanceProfile(
                _autoScalerWrapper.BadCredsProfileName,
                _autoScalerWrapper.BadCredsRoleName
```

```
            );
            await
 _recommendations.DestroyDatabaseByName(_recommendations.TableName);
        }
        else
        {
            Console.WriteLine(
                "Ok, we'll leave the resources intact.\n" +
                "Don't forget to delete them when you're done with them or you
 might incur unexpected charges."
            );
        }

        Console.WriteLine(new string('-', 80));
        return true;
    }
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
/// <summary>
/// Encapsulates Amazon EC2 Auto Scaling and EC2 management methods.
/// </summary>
public class AutoScalerWrapper
{
    private readonly IAmazonAutoScaling _amazonAutoScaling;
    private readonly IAmazonEC2 _amazonEc2;
    private readonly IAmazonSimpleSystemsManagement _amazonSsm;
    private readonly IAmazonIdentityManagementService _amazonIam;
    private readonly ILogger<AutoScalerWrapper> _logger;

    private readonly string _instanceType = "";
    private readonly string _amiParam = "";
    private readonly string _launchTemplateName = "";
    private readonly string _groupName = "";
    private readonly string _instancePolicyName = "";
    private readonly string _instanceRoleName = "";
    private readonly string _instanceProfileName = "";
    private readonly string _badCredsProfileName = "";
    private readonly string _badCredsRoleName = "";
    private readonly string _badCredsPolicyName = "";
    private readonly string _keyPairName = "";
```

```
    public string GroupName => _groupName;
    public string KeyPairName => _keyPairName;
    public string LaunchTemplateName => _launchTemplateName;
    public string InstancePolicyName => _instancePolicyName;
    public string BadCredsProfileName => _badCredsProfileName;
    public string BadCredsRoleName => _badCredsRoleName;
    public string BadCredsPolicyName => _badCredsPolicyName;

    /// <summary>
    /// Constructor for the AutoScalerWrapper.
    /// </summary>
    /// <param name="amazonAutoScaling">The injected AutoScaling client.</param>
    /// <param name="amazonEc2">The injected EC2 client.</param>
    /// <param name="amazonIam">The injected IAM client.</param>
    /// <param name="amazonSsm">The injected SSM client.</param>
    public AutoScalerWrapper(
        IAmazonAutoScaling amazonAutoScaling,
        IAmazonEC2 amazonEc2,
        IAmazonSimpleSystemsManagement amazonSsm,
        IAmazonIdentityManagementService amazonIam,
        IConfiguration configuration,
        ILogger<AutoScalerWrapper> logger)
    {
        _amazonAutoScaling = amazonAutoScaling;
        _amazonEc2 = amazonEc2;
        _amazonSsm = amazonSsm;
        _amazonIam = amazonIam;
        _logger = logger;

        var prefix = configuration["resourcePrefix"];
        _instanceType = configuration["instanceType"];
        _amiParam = configuration["amiParam"];

        _launchTemplateName = prefix + "-template";
        _groupName = prefix + "-group";
        _instancePolicyName = prefix + "-pol";
        _instanceRoleName = prefix + "-role";
        _instanceProfileName = prefix + "-prof";
        _badCredsPolicyName = prefix + "-bc-pol";
        _badCredsRoleName = prefix + "-bc-role";
        _badCredsProfileName = prefix + "-bc-prof";
        _keyPairName = prefix + "-key-pair";
    }
```

```
   /// <summary>
   /// Create a policy, role, and profile that is associated with instances with
a specified name.
   /// An instance's associated profile defines a role that is assumed by the
   /// instance.The role has attached policies that specify the AWS permissions
granted to
   /// clients that run on the instance.
   /// </summary>
   /// <param name="policyName">Name to use for the policy.</param>
   /// <param name="roleName">Name to use for the role.</param>
   /// <param name="profileName">Name to use for the profile.</param>
   /// <param name="ssmOnlyPolicyFile">Path to a policy file for SSM.</param>
   /// <param name="awsManagedPolicies">AWS Managed policies to be attached to
the role.</param>
   /// <returns>The Arn of the profile.</returns>
   public async Task<string> CreateInstanceProfileWithName(
       string policyName,
       string roleName,
       string profileName,
       string ssmOnlyPolicyFile,
       List<string>? awsManagedPolicies = null)
   {

       var assumeRoleDoc = "{" +
                                   "\"Version\": \"2012-10-17\"," +
                                   "\"Statement\": [{" +
                                        "\"Effect\": \"Allow\"," +
                                        "\"Principal\": {" +
                                        "\"Service\": [" +
                                            "\"ec2.amazonaws.com\"" +
                                        "]" +
                                        "}," +
                                   "\"Action\": \"sts:AssumeRole\"" +
                                   "}]" +
                               "}";

       var policyDocument = await File.ReadAllTextAsync(ssmOnlyPolicyFile);

       var policyArn = "";

       try
       {
           var createPolicyResult = await _amazonIam.CreatePolicyAsync(
               new CreatePolicyRequest
```

```
                {
                    PolicyName = policyName,
                    PolicyDocument = policyDocument
                });
            policyArn = createPolicyResult.Policy.Arn;
        }
        catch (EntityAlreadyExistsException)
        {
            // The policy already exists, so we look it up to get the Arn.
            var policiesPaginator = _amazonIam.Paginators.ListPolicies(
                new ListPoliciesRequest()
                {
                    Scope = PolicyScopeType.Local
                });
            // Get the entire list using the paginator.
            await foreach (var policy in policiesPaginator.Policies)
            {
                if (policy.PolicyName.Equals(policyName))
                {
                    policyArn = policy.Arn;
                }
            }

            if (policyArn == null)
            {
                throw new InvalidOperationException("Policy not found");
            }
        }

        try
        {
            await _amazonIam.CreateRoleAsync(new CreateRoleRequest()
            {
                RoleName = roleName,
                AssumeRolePolicyDocument = assumeRoleDoc,
            });
            await _amazonIam.AttachRolePolicyAsync(new AttachRolePolicyRequest()
            {
                RoleName = roleName,
                PolicyArn = policyArn
            });
            if (awsManagedPolicies != null)
            {
                foreach (var awsPolicy in awsManagedPolicies)
```

```
                {
                    await _amazonIam.AttachRolePolicyAsync(new
AttachRolePolicyRequest()
                    {
                        PolicyArn = $"arn:aws:iam::aws:policy/{awsPolicy}",
                        RoleName = roleName
                    });
                }
            }
        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Role already exists.");
        }

        string profileArn = "";
        try
        {
            var profileCreateResponse = await
_amazonIam.CreateInstanceProfileAsync(
                new CreateInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            // Allow time for the profile to be ready.
            profileArn = profileCreateResponse.InstanceProfile.Arn;
            Thread.Sleep(10000);
            await _amazonIam.AddRoleToInstanceProfileAsync(
                new AddRoleToInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });

        }
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine("Policy already exists.");
            var profileGetResponse = await _amazonIam.GetInstanceProfileAsync(
                new GetInstanceProfileRequest()
                {
                    InstanceProfileName = profileName
                });
            profileArn = profileGetResponse.InstanceProfile.Arn;
```

```
        }
        return profileArn;
    }

    /// <summary>
    /// Create a new key pair and save the file.
    /// </summary>
    /// <param name="newKeyPairName">The name of the new key pair.</param>
    /// <returns>Async task.</returns>
    public async Task CreateKeyPair(string newKeyPairName)
    {
        try
        {
            var keyResponse = await _amazonEc2.CreateKeyPairAsync(
                new CreateKeyPairRequest() { KeyName = newKeyPairName });
            await File.WriteAllTextAsync($"{newKeyPairName}.pem",
                keyResponse.KeyPair.KeyMaterial);
            Console.WriteLine($"Created key pair {newKeyPairName}.");
        }
        catch (AlreadyExistsException)
        {
            Console.WriteLine("Key pair already exists.");
        }
    }

    /// <summary>
    /// Delete the key pair and file by name.
    /// </summary>
    /// <param name="deleteKeyPairName">The key pair to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteKeyPairByName(string deleteKeyPairName)
    {
        try
        {
            await _amazonEc2.DeleteKeyPairAsync(
                new DeleteKeyPairRequest() { KeyName = deleteKeyPairName });
            File.Delete($"{deleteKeyPairName}.pem");
        }
        catch (FileNotFoundException)
        {
            Console.WriteLine($"Key pair {deleteKeyPairName} not found.");
        }
    }
```

```
   /// <summary>
   /// Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling.
   /// The launch template specifies a Bash script in its user data field that
runs after
   /// the instance is started. This script installs the Python packages and
starts a Python
   /// web server on the instance.
   /// </summary>
   /// <param name="startupScriptPath">The path to a Bash script file that is
run.</param>
   /// <param name="instancePolicyPath">The path to a permissions policy to
create and attach to the profile.</param>
   /// <returns>The template object.</returns>
   public async Task<Amazon.EC2.Model.LaunchTemplate> CreateTemplate(string
startupScriptPath, string instancePolicyPath)
   {
       try
       {
           await CreateKeyPair(_keyPairName);
           await CreateInstanceProfileWithName(_instancePolicyName,
_instanceRoleName,
               _instanceProfileName, instancePolicyPath);

           var startServerText = await File.ReadAllTextAsync(startupScriptPath);
           var plainTextBytes =
System.Text.Encoding.UTF8.GetBytes(startServerText);

           var amiLatest = await _amazonSsm.GetParameterAsync(
               new GetParameterRequest() { Name = _amiParam });
           var amiId = amiLatest.Parameter.Value;
           var launchTemplateResponse = await
_amazonEc2.CreateLaunchTemplateAsync(
               new CreateLaunchTemplateRequest()
               {
                   LaunchTemplateName = _launchTemplateName,
                   LaunchTemplateData = new RequestLaunchTemplateData()
                   {
                       InstanceType = _instanceType,
                       ImageId = amiId,
                       IamInstanceProfile =
                           new

LaunchTemplateIamInstanceProfileSpecificationRequest()
```

```
                            {
                                Name = _instanceProfileName
                            },
                        KeyName = _keyPairName,
                        UserData = System.Convert.ToBase64String(plainTextBytes)
                    }
                });
            return launchTemplateResponse.LaunchTemplate;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.AlreadyExistsException")
            {
                _logger.LogError($"Could not create the template, the name
{_launchTemplateName} already exists. " +
                                 $"Please try again with a unique name.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while creating the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Get a list of Availability Zones in the AWS Region of the Amazon EC2
Client.
    /// </summary>
    /// <returns>A list of availability zones.</returns>
    public async Task<List<string>> DescribeAvailabilityZones()
    {
        try
        {
            var zoneResponse = await _amazonEc2.DescribeAvailabilityZonesAsync(
                new DescribeAvailabilityZonesRequest());
            return zoneResponse.AvailabilityZones.Select(z =>
z.ZoneName).ToList();
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            _logger.LogError($"An Amazon EC2 error occurred while listing
 availability zones.: {ec2Exception.Message}");
            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while listing availability
 zones.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Create an EC2 Auto Scaling group of a specified size and name.
    /// </summary>
    /// <param name="groupSize">The size for the group.</param>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="availabilityZones">The availability zones for the group.</
param>
    /// <returns>Async task.</returns>
    public async Task CreateGroupOfSize(int groupSize, string groupName,
 List<string> availabilityZones)
    {
        try
        {
            await _amazonAutoScaling.CreateAutoScalingGroupAsync(
                new CreateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    AvailabilityZones = availabilityZones,
                    LaunchTemplate =
                        new
 Amazon.AutoScaling.Model.LaunchTemplateSpecification()
                        {
                            LaunchTemplateName = _launchTemplateName,
                            Version = "$Default"
                        },
                    MaxSize = groupSize,
                    MinSize = groupSize
                });
            Console.WriteLine($"Created EC2 Auto Scaling group {groupName} with
 size {groupSize}.");
        }
```

```
        catch (EntityAlreadyExistsException)
        {
            Console.WriteLine($"EC2 Auto Scaling group {groupName} already
exists.");
        }
    }

    /// <summary>
    /// Get the default VPC for the account.
    /// </summary>
    /// <returns>The default VPC object.</returns>
    public async Task<Vpc> GetDefaultVpc()
    {
        try
        {
            var vpcResponse = await _amazonEc2.DescribeVpcsAsync(
                new DescribeVpcsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("is-default", new List<string>() { "true" })
                    }
                });
            return vpcResponse.Vpcs[0];
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "UnauthorizedOperation")
            {
                _logger.LogError(ec2Exception, $"You do not have the necessary
permissions to describe VPCs.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the vpcs.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
```

```
    /// Get all the subnets for a Vpc in a set of availability zones.
    /// </summary>
    /// <param name="vpcId">The Id of the Vpc.</param>
    /// <param name="availabilityZones">The list of availability zones.</param>
    /// <returns>The collection of subnet objects.</returns>
    public async Task<List<Subnet>> GetAllVpcSubnetsForZones(string vpcId,
List<string> availabilityZones)
    {
        try
        {
            var subnets = new List<Subnet>();
            var subnetPaginator = _amazonEc2.Paginators.DescribeSubnets(
                new DescribeSubnetsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("vpc-id", new List<string>() { vpcId }),
                        new("availability-zone", availabilityZones),
                        new("default-for-az", new List<string>() { "true" })
                    }
                });

            // Get the entire list using the paginator.
            await foreach (var subnet in subnetPaginator.Subnets)
            {
                subnets.Add(subnet);
            }

            return subnets;
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidVpcID.NotFound")
            {
                _logger.LogError(ec2Exception, $"The specified VPC ID {vpcId}
does not exist.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while describing the
subnets.: {ex.Message}");
```

```
            throw;
        }
    }

    /// <summary>
    /// Delete a launch template by name.
    /// </summary>
    /// <param name="templateName">The name of the template to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTemplateByName(string templateName)
    {
        try
        {
            await _amazonEc2.DeleteLaunchTemplateAsync(
                new DeleteLaunchTemplateRequest()
                {
                    LaunchTemplateName = templateName
                });
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode ==
"InvalidLaunchTemplateName.NotFoundException")
            {
                _logger.LogError(
                    $"Could not delete the template, the name
{_launchTemplateName} was not found.");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError($"An error occurred while deleting the template.:
{ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Detaches a role from an instance profile, detaches policies from the
role,
    /// and deletes all the resources.
    /// </summary>
```

```
    /// <param name="profileName">The name of the profile to delete.</param>
    /// <param name="roleName">The name of the role to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteInstanceProfile(string profileName, string roleName)
    {
        try
        {
            await _amazonIam.RemoveRoleFromInstanceProfileAsync(
                new RemoveRoleFromInstanceProfileRequest()
                {
                    InstanceProfileName = profileName,
                    RoleName = roleName
                });
            await _amazonIam.DeleteInstanceProfileAsync(
                new DeleteInstanceProfileRequest() { InstanceProfileName =
profileName });
            var attachedPolicies = await
_amazonIam.ListAttachedRolePoliciesAsync(
                new ListAttachedRolePoliciesRequest() { RoleName = roleName });
            foreach (var policy in attachedPolicies.AttachedPolicies)
            {
                await _amazonIam.DetachRolePolicyAsync(
                    new DetachRolePolicyRequest()
                    {
                        RoleName = roleName,
                        PolicyArn = policy.PolicyArn
                    });
                // Delete the custom policies only.
                if (!policy.PolicyArn.StartsWith("arn:aws:iam::aws"))
                {
                    await _amazonIam.DeletePolicyAsync(
                        new Amazon.IdentityManagement.Model.DeletePolicyRequest()
                        {
                            PolicyArn = policy.PolicyArn
                        });
                }
            }

            await _amazonIam.DeleteRoleAsync(
                new DeleteRoleRequest() { RoleName = roleName });
        }
        catch (NoSuchEntityException)
        {
            Console.WriteLine($"Instance profile {profileName} does not exist.");
```

```
        }
    }

    /// <summary>
    /// Gets data about the instances in an EC2 Auto Scaling group by its group
name.
    /// </summary>
    /// <param name="group">The name of the auto scaling group.</param>
    /// <returns>A collection of instance Ids.</returns>
    public async Task<IEnumerable<string>> GetInstancesByGroupName(string group)
    {
        var instanceResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { group }
            });
        var instanceIds = instanceResponse.AutoScalingGroups.SelectMany(
            g => g.Instances.Select(i => i.InstanceId));
        return instanceIds;
    }

    /// <summary>
    /// Get the instance profile association data for an instance.
    /// </summary>
    /// <param name="instanceId">The Id of the instance.</param>
    /// <returns>Instance profile associations data.</returns>
    public async Task<IamInstanceProfileAssociation> GetInstanceProfile(string
instanceId)
    {
        try
        {
            var response = await
_amazonEc2.DescribeIamInstanceProfileAssociationsAsync(
                new DescribeIamInstanceProfileAssociationsRequest()
                {
                    Filters = new List<Amazon.EC2.Model.Filter>()
                    {
                        new("instance-id", new List<string>() { instanceId })
                    },
                });
            return response.IamInstanceProfileAssociations[0];
        }
        catch (AmazonEC2Exception ec2Exception)
```

```
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while creating the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Replace the profile associated with a running instance. After the profile
is replaced, the instance
    /// is rebooted to ensure that it uses the new profile. When the instance is
ready, Systems Manager is
    /// used to restart the Python web server.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to update.</param>
    /// <param name="credsProfileName">The name of the new profile to associate
with the specified instance.</param>
    /// <param name="associationId">The Id of the existing profile association
for the instance.</param>
    /// <returns>Async task.</returns>
    public async Task ReplaceInstanceProfile(string instanceId, string
credsProfileName, string associationId)
    {
        try
        {
            await _amazonEc2.ReplaceIamInstanceProfileAssociationAsync(
                new ReplaceIamInstanceProfileAssociationRequest()
                {
                    AssociationId = associationId,
                    IamInstanceProfile = new IamInstanceProfileSpecification()
                    {
                        Name = credsProfileName
                    }
                });
```

```
            // Allow time before resetting.
            Thread.Sleep(25000);

            await _amazonEc2.RebootInstancesAsync(
                new RebootInstancesRequest(new List<string>() { instanceId }));
            Thread.Sleep(25000);
            var instanceReady = false;
            var retries = 5;
            while (retries-- > 0 && !instanceReady)
            {
                var instancesPaginator =
                    _amazonSsm.Paginators.DescribeInstanceInformation(
                        new DescribeInstanceInformationRequest());
                // Get the entire list using the paginator.
                await foreach (var instance in
instancesPaginator.InstanceInformationList)
                {
                    instanceReady = instance.InstanceId == instanceId;
                    if (instanceReady)
                    {
                        break;
                    }
                }
            }
            Console.WriteLine("Waiting for instance to be running.");
            await WaitForInstanceState(instanceId, InstanceStateName.Running);
            Console.WriteLine("Instance ready.");
            Console.WriteLine($"Sending restart command to instance
{instanceId}");
            await _amazonSsm.SendCommandAsync(
                new SendCommandRequest()
                {
                    InstanceIds = new List<string>() { instanceId },
                    DocumentName = "AWS-RunShellScript",
                    Parameters = new Dictionary<string, List<string>>()
                    {
                        {
                            "commands",
                            new List<string>() { "cd / && sudo python3 server.py
80" }
                        }
                    }
                });
```

```
            Console.WriteLine($"Restarted the web server on instance
{instanceId}");
        }
        catch (AmazonEC2Exception ec2Exception)
        {
            if (ec2Exception.ErrorCode == "InvalidInstanceID.NotFound")
            {
                _logger.LogError(ec2Exception, $"Instance {instanceId} not
found");
            }

            throw;
        }
        catch (Exception ex)
        {
            _logger.LogError(ex, $"An error occurred while replacing the
template.: {ex.Message}");
            throw;
        }
    }

    /// <summary>
    /// Try to terminate an instance by its Id.
    /// </summary>
    /// <param name="instanceId">The Id of the instance to terminate.</param>
    /// <returns>Async task.</returns>
    public async Task TryTerminateInstanceById(string instanceId)
    {
        var stopping = false;
        Console.WriteLine($"Stopping {instanceId}...");
        while (!stopping)
        {
            try
            {
                await
_amazonAutoScaling.TerminateInstanceInAutoScalingGroupAsync(
                    new TerminateInstanceInAutoScalingGroupRequest()
                    {
                        InstanceId = instanceId,
                        ShouldDecrementDesiredCapacity = false
                    });
                stopping = true;
            }
            catch (ScalingActivityInProgressException)
```

```
            {
                Console.WriteLine($"Scaling activity in progress for
{instanceId}. Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Tries to delete the EC2 Auto Scaling group. If the group is in use or in
progress,
    /// waits and retries until the group is successfully deleted.
    /// </summary>
    /// <param name="groupName">The name of the group to try to delete.</param>
    /// <returns>Async task.</returns>
    public async Task TryDeleteGroupByName(string groupName)
    {
        var stopped = false;
        while (!stopped)
        {
            try
            {
                await _amazonAutoScaling.DeleteAutoScalingGroupAsync(
                    new DeleteAutoScalingGroupRequest()
                    {
                        AutoScalingGroupName = groupName
                    });
                stopped = true;
            }
            catch (Exception e)
                when ((e is ScalingActivityInProgressException)
                      || (e is Amazon.AutoScaling.Model.ResourceInUseException))
            {
                Console.WriteLine($"Some instances are still running.
Waiting...");
                Thread.Sleep(10000);
            }
        }
    }

    /// <summary>
    /// Terminate instances and delete the Auto Scaling group by name.
    /// </summary>
    /// <param name="groupName">The name of the group to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task TerminateAndDeleteAutoScalingGroupWithName(string
groupName)
    {
        var describeGroupsResponse = await
_amazonAutoScaling.DescribeAutoScalingGroupsAsync(
            new DescribeAutoScalingGroupsRequest()
            {
                AutoScalingGroupNames = new List<string>() { groupName }
            });
        if (describeGroupsResponse.AutoScalingGroups.Any())
        {
            // Update the size to 0.
            await _amazonAutoScaling.UpdateAutoScalingGroupAsync(
                new UpdateAutoScalingGroupRequest()
                {
                    AutoScalingGroupName = groupName,
                    MinSize = 0
                });
            var group = describeGroupsResponse.AutoScalingGroups[0];
            foreach (var instance in group.Instances)
            {
                await TryTerminateInstanceById(instance.InstanceId);
            }

            await TryDeleteGroupByName(groupName);
        }
        else
        {
            Console.WriteLine($"No groups found with name {groupName}.");
        }
    }


    /// <summary>
    /// Get the default security group for a specified Vpc.
    /// </summary>
    /// <param name="vpc">The Vpc to search.</param>
    /// <returns>The default security group.</returns>
    public async Task<SecurityGroup> GetDefaultSecurityGroupForVpc(Vpc vpc)
    {
        var groupResponse = await _amazonEc2.DescribeSecurityGroupsAsync(
            new DescribeSecurityGroupsRequest()
            {
```

```
                Filters = new List<Amazon.EC2.Model.Filter>()
                {
                    new ("group-name", new List<string>() { "default" }),
                    new ("vpc-id", new List<string>() { vpc.VpcId })
                }
            });
        return groupResponse.SecurityGroups[0];
    }

    /// <summary>
    /// Verify the default security group of a Vpc allows ingress from the
calling computer.
    /// This can be done by allowing ingress from this computer's IP address.
    /// In some situations, such as connecting from a corporate network, you must
instead specify
    /// a prefix list Id. You can also temporarily open the port to any IP
address while running this example.
    /// If you do, be sure to remove public access when you're done.
    /// </summary>
    /// <param name="vpc">The group to check.</param>
    /// <param name="port">The port to verify.</param>
    /// <param name="ipAddress">This computer's IP address.</param>
    /// <returns>True if the ip address is allowed on the group.</returns>
    public bool VerifyInboundPortForGroup(SecurityGroup group, int port, string
ipAddress)
    {
        var portIsOpen = false;
        foreach (var ipPermission in group.IpPermissions)
        {
            if (ipPermission.FromPort == port)
            {
                foreach (var ipRange in ipPermission.Ipv4Ranges)
                {
                    var cidr = ipRange.CidrIp;
                    if (cidr.StartsWith(ipAddress) || cidr == "0.0.0.0/0")
                    {
                        portIsOpen = true;
                    }
                }

                if (ipPermission.PrefixListIds.Any())
                {
                    portIsOpen = true;
                }
```

```
                if (!portIsOpen)
                {
                    Console.WriteLine("The inbound rule does not appear to be
open to either this computer's IP\n" +
                                      "address, to all IP addresses (0.0.0.0/0),
or to a prefix list ID.");
                }
                else
                {
                    break;
                }
            }
        }

        return portIsOpen;
    }

    /// <summary>
    /// Add an ingress rule to the specified security group that allows access on
the
    /// specified port from the specified IP address.
    /// </summary>
    /// <param name="groupId">The Id of the security group to modify.</param>
    /// <param name="port">The port to open.</param>
    /// <param name="ipAddress">The IP address to allow access.</param>
    /// <returns>Async task.</returns>
    public async Task OpenInboundPort(string groupId, int port, string ipAddress)
    {
        await _amazonEc2.AuthorizeSecurityGroupIngressAsync(
            new AuthorizeSecurityGroupIngressRequest()
            {
                GroupId = groupId,
                IpPermissions = new List<IpPermission>()
                {
                    new IpPermission()
                    {
                        FromPort = port,
                        ToPort = port,
                        IpProtocol = "tcp",
                        Ipv4Ranges = new List<IpRange>()
                        {
                            new IpRange() { CidrIp = $"{ipAddress}/32" }
                        }
```

```
                    }
                }
            });
    }

    /// <summary>
    /// Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
 Scaling group.
    /// The
    /// </summary>
    /// <param name="autoScalingGroupName">The name of the Auto Scaling group.</
param>
    /// <param name="targetGroupArn">The Arn for the target group.</param>
    /// <returns>Async task.</returns>
    public async Task AttachLoadBalancerToGroup(string autoScalingGroupName,
 string targetGroupArn)
    {
        await _amazonAutoScaling.AttachLoadBalancerTargetGroupsAsync(
            new AttachLoadBalancerTargetGroupsRequest()
            {
                AutoScalingGroupName = autoScalingGroupName,
                TargetGroupARNs = new List<string>() { targetGroupArn }
            });
    }

    /// <summary>
    /// Wait until an EC2 instance is in a specified state.
    /// </summary>
    /// <param name="instanceId">The instance Id.</param>
    /// <param name="stateName">The state to wait for.</param>
    /// <returns>A Boolean value indicating the success of the action.</returns>
    public async Task<bool> WaitForInstanceState(string instanceId,
 InstanceStateName stateName)
    {
        var request = new DescribeInstancesRequest
        {
            InstanceIds = new List<string> { instanceId }
        };

        // Wait until the instance is in the specified state.
        var hasState = false;
        do
        {
            // Wait 5 seconds.
```

```
            Thread.Sleep(5000);

            // Check for the desired state.
            var response = await _amazonEc2.DescribeInstancesAsync(request);
            var instance = response.Reservations[0].Instances[0];
            hasState = instance.State.Name == stateName;
            Console.Write(". ");
        } while (!hasState);

        return hasState;
    }
}
```

创建一个包含弹性负载均衡操作的类。

```
/// <summary>
/// Encapsulates Elastic Load Balancer actions.
/// </summary>
public class ElasticLoadBalancerWrapper
{
    private readonly IAmazonElasticLoadBalancingV2 _amazonElasticLoadBalancingV2;
    private string? _endpoint = null;
    private readonly string _targetGroupName = "";
    private readonly string _loadBalancerName = "";
    HttpClient _httpClient = new();

    public string TargetGroupName => _targetGroupName;
    public string LoadBalancerName => _loadBalancerName;

    /// <summary>
    /// Constructor for the Elastic Load Balancer wrapper.
    /// </summary>
    /// <param name="amazonElasticLoadBalancingV2">The injected load balancing v2
 client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public ElasticLoadBalancerWrapper(
        IAmazonElasticLoadBalancingV2 amazonElasticLoadBalancingV2,
        IConfiguration configuration)
    {
        _amazonElasticLoadBalancingV2 = amazonElasticLoadBalancingV2;
        var prefix = configuration["resourcePrefix"];
```

```
        _targetGroupName = prefix + "-tg";
        _loadBalancerName = prefix + "-lb";
    }

    /// <summary>
    /// Get the HTTP Endpoint of a load balancer by its name.
    /// </summary>
    /// <param name="loadBalancerName">The name of the load balancer.</param>
    /// <returns>The HTTP endpoint.</returns>
    public async Task<string> GetEndpointForLoadBalancerByName(string
loadBalancerName)
    {
        if (_endpoint == null)
        {
            var endpointResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { loadBalancerName }
                    });
            _endpoint = endpointResponse.LoadBalancers[0].DNSName;
        }

        return _endpoint;
    }

    /// <summary>
    /// Return the GET response for an endpoint as text.
    /// </summary>
    /// <param name="endpoint">The endpoint for the request.</param>
    /// <returns>The request response.</returns>
    public async Task<string> GetEndPointResponse(string endpoint)
    {
        var endpointResponse = await _httpClient.GetAsync($"http://{endpoint}");
        var textResponse = await endpointResponse.Content.ReadAsStringAsync();
        return textResponse!;
    }

    /// <summary>
    /// Get the target health for a group by name.
    /// </summary>
    /// <param name="groupName">The name of the group.</param>
    /// <returns>The collection of health descriptions.</returns>
```

```
    public async Task<List<TargetHealthDescription>>
 CheckTargetHealthForGroup(string groupName)
    {
        List<TargetHealthDescription> result = null!;
        try
        {
            var groupResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                    new DescribeTargetGroupsRequest()
                    {
                        Names = new List<string>() { groupName }
                    });
            var healthResponse =
                await _amazonElasticLoadBalancingV2.DescribeTargetHealthAsync(
                    new DescribeTargetHealthRequest()
                    {
                        TargetGroupArn =
 groupResponse.TargetGroups[0].TargetGroupArn
                    });
            ;
            result = healthResponse.TargetHealthDescriptions;
        }
        catch (TargetGroupNotFoundException)
        {
            Console.WriteLine($"Target group {groupName} not found.");
        }
        return result;
    }

    /// <summary>
    /// Create an Elastic Load Balancing target group. The target group specifies
 how the load balancer forwards
    /// requests to instances in the group and how instance health is checked.
    ///
    /// To speed up this demo, the health check is configured with shortened
 times and lower thresholds. In production,
    /// you might want to decrease the sensitivity of your health checks to avoid
 unwanted failures.
    /// </summary>
    /// <param name="groupName">The name for the group.</param>
    /// <param name="protocol">The protocol, such as HTTP.</param>
    /// <param name="port">The port to use to forward requests, such as 80.</
param>
```

```
   /// <param name="vpcId">The Id of the Vpc in which the load balancer
exists.</param>
   /// <returns>The new TargetGroup object.</returns>
   public async Task<TargetGroup> CreateTargetGroupOnVpc(string groupName,
ProtocolEnum protocol, int port, string vpcId)
   {
       var createResponse = await
_amazonElasticLoadBalancingV2.CreateTargetGroupAsync(
           new CreateTargetGroupRequest()
           {
               Name = groupName,
               Protocol = protocol,
               Port = port,
               HealthCheckPath = "/healthcheck",
               HealthCheckIntervalSeconds = 10,
               HealthCheckTimeoutSeconds = 5,
               HealthyThresholdCount = 2,
               UnhealthyThresholdCount = 2,
               VpcId = vpcId
           });
       var targetGroup = createResponse.TargetGroups[0];
       return targetGroup;
   }

   /// <summary>
   /// Create an Elastic Load Balancing load balancer that uses the specified
subnets
   /// and forwards requests to the specified target group.
   /// </summary>
   /// <param name="name">The name for the new load balancer.</param>
   /// <param name="subnetIds">Subnets for the load balancer.</param>
   /// <param name="targetGroup">Target group for forwarded requests.</param>
   /// <returns>The new LoadBalancer object.</returns>
   public async Task<LoadBalancer> CreateLoadBalancerAndListener(string name,
List<string> subnetIds, TargetGroup targetGroup)
   {
       var createLbResponse = await
_amazonElasticLoadBalancingV2.CreateLoadBalancerAsync(
           new CreateLoadBalancerRequest()
           {
               Name = name,
               Subnets = subnetIds
           });
       var loadBalancerArn = createLbResponse.LoadBalancers[0].LoadBalancerArn;
```

```
        // Wait for load balancer to be available.
        var loadBalancerReady = false;
        while (!loadBalancerReady)
        {
            try
            {
                var describeResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                        new DescribeLoadBalancersRequest()
                        {
                            Names = new List<string>() { name }
                        });

                var loadBalancerState =
describeResponse.LoadBalancers[0].State.Code;

                loadBalancerReady = loadBalancerState ==
LoadBalancerStateEnum.Active;
            }
            catch (LoadBalancerNotFoundException)
            {
                loadBalancerReady = false;
            }
            Thread.Sleep(10000);
        }
        // Create the listener.
        await _amazonElasticLoadBalancingV2.CreateListenerAsync(
            new CreateListenerRequest()
            {
                LoadBalancerArn = loadBalancerArn,
                Protocol = targetGroup.Protocol,
                Port = targetGroup.Port,
                DefaultActions = new List<Action>()
                {
                    new Action()
                    {
                        Type = ActionTypeEnum.Forward,
                        TargetGroupArn = targetGroup.TargetGroupArn
                    }
                }
            });
        return createLbResponse.LoadBalancers[0];
```

```
    }

    /// <summary>
    /// Verify this computer can successfully send a GET request to the
    /// load balancer endpoint.
    /// </summary>
    /// <param name="endpoint">The endpoint to check.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> VerifyLoadBalancerEndpoint(string endpoint)
    {
        var success = false;
        var retries = 3;
        while (!success && retries > 0)
        {
            try
            {
                var endpointResponse = await _httpClient.GetAsync($"http://
{endpoint}");
                Console.WriteLine($"Response: {endpointResponse.StatusCode}.");

                if (endpointResponse.IsSuccessStatusCode)
                {
                    success = true;
                }
                else
                {
                    retries = 0;
                }
            }
            catch (HttpRequestException)
            {
                Console.WriteLine("Connection error, retrying...");
                retries--;
                Thread.Sleep(10000);
            }
        }

        return success;
    }

    /// <summary>
    /// Delete a load balancer by its specified name.
    /// </summary>
    /// <param name="name">The name of the load balancer to delete.</param>
```

```
    /// <returns>Async task.</returns>
    public async Task DeleteLoadBalancerByName(string name)
    {
        try
        {
            var describeLoadBalancerResponse =
                await _amazonElasticLoadBalancingV2.DescribeLoadBalancersAsync(
                    new DescribeLoadBalancersRequest()
                    {
                        Names = new List<string>() { name }
                    });
            var lbArn =
describeLoadBalancerResponse.LoadBalancers[0].LoadBalancerArn;
            await _amazonElasticLoadBalancingV2.DeleteLoadBalancerAsync(
                new DeleteLoadBalancerRequest()
                {
                    LoadBalancerArn = lbArn
                }
            );
        }
        catch (LoadBalancerNotFoundException)
        {
            Console.WriteLine($"Load balancer {name} not found.");
        }
    }

    /// <summary>
    /// Delete a TargetGroup by its specified name.
    /// </summary>
    /// <param name="groupName">Name of the group to delete.</param>
    /// <returns>Async task.</returns>
    public async Task DeleteTargetGroupByName(string groupName)
    {
        var done = false;
        while (!done)
        {
            try
            {
                var groupResponse =
                    await
_amazonElasticLoadBalancingV2.DescribeTargetGroupsAsync(
                        new DescribeTargetGroupsRequest()
                        {
                            Names = new List<string>() { groupName }
```

```
                    });

                var targetArn = groupResponse.TargetGroups[0].TargetGroupArn;
                await _amazonElasticLoadBalancingV2.DeleteTargetGroupAsync(
                    new DeleteTargetGroupRequest() { TargetGroupArn =
 targetArn });
                Console.WriteLine($"Deleted load balancing target group
 {groupName}.");
                done = true;
            }
            catch (TargetGroupNotFoundException)
            {
                Console.WriteLine(
                    $"Target group {groupName} not found, could not delete.");
                done = true;
            }
            catch (ResourceInUseException)
            {
                Console.WriteLine("Target group not yet released, waiting...");
                Thread.Sleep(10000);
            }
        }
    }
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
/// <summary>
/// Encapsulates a DynamoDB table to use as a service that recommends books,
 movies, and songs.
/// </summary>
public class Recommendations
{
    private readonly IAmazonDynamoDB _amazonDynamoDb;
    private readonly DynamoDBContext _context;
    private readonly string _tableName;

    public string TableName => _tableName;

    /// <summary>
    /// Constructor for the Recommendations service.
    /// </summary>
```

```
    /// <param name="amazonDynamoDb">The injected DynamoDb client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public Recommendations(IAmazonDynamoDB amazonDynamoDb, IConfiguration
configuration)
    {
        _amazonDynamoDb = amazonDynamoDb;
        _context = new DynamoDBContext(_amazonDynamoDb);
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
    /// Create the DynamoDb table with a specified name.
    /// </summary>
    /// <param name="tableName">The name for the table.</param>
    /// <returns>True when ready.</returns>
    public async Task<bool> CreateDatabaseWithName(string tableName)
    {
        try
        {
            Console.Write($"Creating table {tableName}...");
            var createRequest = new CreateTableRequest()
            {
                TableName = tableName,
                AttributeDefinitions = new List<AttributeDefinition>()
                    {
                        new AttributeDefinition()
                        {
                            AttributeName = "MediaType",
                            AttributeType = ScalarAttributeType.S
                        },
                        new AttributeDefinition()
                        {
                            AttributeName = "ItemId",
                            AttributeType = ScalarAttributeType.N
                        }
                    },
                KeySchema = new List<KeySchemaElement>()
                    {
                        new KeySchemaElement()
                        {
                            AttributeName = "MediaType",
                            KeyType = KeyType.HASH
                        },
                        new KeySchemaElement()
```

```
                        {
                            AttributeName = "ItemId",
                            KeyType = KeyType.RANGE
                        }
                    },
                ProvisionedThroughput = new ProvisionedThroughput()
                {
                    ReadCapacityUnits = 5,
                    WriteCapacityUnits = 5
                }
            };
            await _amazonDynamoDb.CreateTableAsync(createRequest);

            // Wait until the table is ACTIVE and then report success.
            Console.Write("\nWaiting for table to become active...");

            var request = new DescribeTableRequest
            {
                TableName = tableName
            };

            TableStatus status;
            do
            {
                Thread.Sleep(2000);

                var describeTableResponse = await
_amazonDynamoDb.DescribeTableAsync(request);
                status = describeTableResponse.Table.TableStatus;

                Console.Write(".");
            }
            while (status != "ACTIVE");

            return status == TableStatus.ACTIVE;
        }
        catch (ResourceInUseException)
        {
            Console.WriteLine($"Table {tableName} already exists.");
            return false;
        }
    }

    /// <summary>
```

```
    /// Populate the database table with data from a specified path.
    /// </summary>
    /// <param name="databaseTableName">The name of the table.</param>
    /// <param name="recommendationsPath">The path of the recommendations data.</
param>
    /// <returns>Async task.</returns>
    public async Task PopulateDatabase(string databaseTableName, string
 recommendationsPath)
    {
        var recommendationsText = await
 File.ReadAllTextAsync(recommendationsPath);
        var records =

 JsonSerializer.Deserialize<RecommendationModel[]>(recommendationsText);
        var batchWrite = _context.CreateBatchWrite<RecommendationModel>();

        foreach (var record in records!)
        {
            batchWrite.AddPutItem(record);
        }

        await batchWrite.ExecuteAsync();
    }

    /// <summary>
    /// Delete the recommendation table by name.
    /// </summary>
    /// <param name="tableName">The name of the recommendation table.</param>
    /// <returns>Async task.</returns>
    public async Task DestroyDatabaseByName(string tableName)
    {
        try
        {
            await _amazonDynamoDb.DeleteTableAsync(
                new DeleteTableRequest() { TableName = tableName });
            Console.WriteLine($"Table {tableName} was deleted.");
        }
        catch (ResourceNotFoundException)
        {
            Console.WriteLine($"Table {tableName} not found");
        }
    }
}
```

创建一个包含 Systems Manager 操作的类。

```
/// <summary>
/// Encapsulates Systems Manager parameter operations. This example uses these
 parameters
/// to drive the demonstration of resilient architecture, such as failure of a
 dependency or
/// how the service responds to a health check.
/// </summary>
public class SmParameterWrapper
{
    private readonly IAmazonSimpleSystemsManagement
 _amazonSimpleSystemsManagement;

    private readonly string _tableParameter = "doc-example-resilient-
architecture-table";
    private readonly string _failureResponseParameter = "doc-example-resilient-
architecture-failure-response";
    private readonly string _healthCheckParameter = "doc-example-resilient-
architecture-health-check";
    private readonly string _tableName = "";

    public string TableParameter => _tableParameter;
    public string TableName => _tableName;
    public string HealthCheckParameter => _healthCheckParameter;
    public string FailureResponseParameter => _failureResponseParameter;

    /// <summary>
    /// Constructor for the SmParameterWrapper.
    /// </summary>
    /// <param name="amazonSimpleSystemsManagement">The injected Simple Systems
 Management client.</param>
    /// <param name="configuration">The injected configuration.</param>
    public SmParameterWrapper(IAmazonSimpleSystemsManagement
 amazonSimpleSystemsManagement, IConfiguration configuration)
    {
        _amazonSimpleSystemsManagement = amazonSimpleSystemsManagement;
        _tableName = configuration["databaseName"]!;
    }

    /// <summary>
```

```
    /// Reset the Systems Manager parameters to starting values for the demo.
    /// </summary>
    /// <returns>Async task.</returns>
    public async Task Reset()
    {
        await this.PutParameterByName(_tableParameter, _tableName);
        await this.PutParameterByName(_failureResponseParameter, "none");
        await this.PutParameterByName(_healthCheckParameter, "shallow");
    }

    /// <summary>
    /// Set the value of a named Systems Manager parameter.
    /// </summary>
    /// <param name="name">The name of the parameter.</param>
    /// <param name="value">The value to set.</param>
    /// <returns>Async task.</returns>
    public async Task PutParameterByName(string name, string value)
    {
        await _amazonSimpleSystemsManagement.PutParameterAsync(
            new PutParameterRequest() { Name = name, Value = value, Overwrite =
 true });
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer

- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
public class Main {

    public static final String fileName = "C:\\AWS\\resworkflow\
\recommendations.json"; // Modify file location.
    public static final String tableName = "doc-example-recommendation-service";
    public static final String startScript = "C:\\AWS\\resworkflow\
\server_startup_script.sh"; // Modify file location.
    public static final String policyFile = "C:\\AWS\\resworkflow\
\instance_policy.json"; // Modify file location.
```

```
    public static final String ssmJSON = "C:\\AWS\\resworkflow\
\ssm_only_policy.json"; // Modify file location.
    public static final String failureResponse = "doc-example-resilient-
architecture-failure-response";
    public static final String healthCheck = "doc-example-resilient-architecture-
health-check";
    public static final String templateName = "doc-example-resilience-template";
    public static final String roleName = "doc-example-resilience-role";
    public static final String policyName = "doc-example-resilience-pol";
    public static final String profileName = "doc-example-resilience-prof";

    public static final String badCredsProfileName = "doc-example-resilience-
prof-bc";

    public static final String targetGroupName = "doc-example-resilience-tg";
    public static final String autoScalingGroupName = "doc-example-resilience-
group";
    public static final String lbName = "doc-example-resilience-lb";
    public static final String protocol = "HTTP";
    public static final int port = 80;

    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws IOException,
 InterruptedException {
        Scanner in = new Scanner(System.in);
        Database database = new Database();
        AutoScaler autoScaler = new AutoScaler();
        LoadBalancer loadBalancer = new LoadBalancer();

        System.out.println(DASHES);
        System.out.println("Welcome to the demonstration of How to Build and
 Manage a Resilient Service!");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("A - SETUP THE RESOURCES");
        System.out.println("Press Enter when you're ready to start deploying
 resources.");
        in.nextLine();
        deploy(loadBalancer);
        System.out.println(DASHES);
        System.out.println(DASHES);
```

```
        System.out.println("B - DEMO THE RESILIENCE FUNCTIONALITY");
        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        demo(loadBalancer);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("C - DELETE THE RESOURCES");
        System.out.println("""
                This concludes the demo of how to build and manage a resilient
service.
                To keep things tidy and to avoid unwanted charges on your
account, we can clean up all AWS resources
                that were created for this demo.
                """);

        System.out.println("\n Do you want to delete the resources (y/n)? ");
        String userInput = in.nextLine().trim().toLowerCase(); // Capture user
input

        if (userInput.equals("y")) {
            // Delete resources here
            deleteResources(loadBalancer, autoScaler, database);
            System.out.println("Resources deleted.");
        } else {
            System.out.println("""
                    Okay, we'll leave the resources intact.
                    Don't forget to delete them when you're done with them or you
might incur unexpected charges.
                    """);
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("The example has completed. ");
        System.out.println("\n Thanks for watching!");
        System.out.println(DASHES);
    }

    // Deletes the AWS resources used in this example.
    private static void deleteResources(LoadBalancer loadBalancer, AutoScaler
autoScaler, Database database)
            throws IOException, InterruptedException {
        loadBalancer.deleteLoadBalancer(lbName);
```

```
        System.out.println("*** Wait 30 secs for resource to be deleted");
        TimeUnit.SECONDS.sleep(30);
        loadBalancer.deleteTargetGroup(targetGroupName);
        autoScaler.deleteAutoScaleGroup(autoScalingGroupName);
        autoScaler.deleteRolesPolicies(policyName, roleName, profileName);
        autoScaler.deleteTemplate(templateName);
        database.deleteTable(tableName);
    }

    private static void deploy(LoadBalancer loadBalancer) throws
InterruptedException, IOException {
        Scanner in = new Scanner(System.in);
        System.out.println(
                """
                        For this demo, we'll use the AWS SDK for Java (v2) to
create several AWS resources
                        to set up a load-balanced web service endpoint and
explore some ways to make it resilient
                        against various kinds of failures.

                        Some of the resources create by this demo are:
                        \t* A DynamoDB table that the web service depends on to
provide book, movie, and song recommendations.
                        \t* An EC2 launch template that defines EC2 instances
that each contain a Python web server.
                        \t* An EC2 Auto Scaling group that manages EC2 instances
across several Availability Zones.
                        \t* An Elastic Load Balancing (ELB) load balancer that
targets the Auto Scaling group to distribute requests.
                        """);

        System.out.println("Press Enter when you're ready.");
        in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("Creating and populating a DynamoDB table named " +
tableName);
        Database database = new Database();
        database.createTable(tableName, fileName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
```

```
                Creating an EC2 launch template that runs '{startup_script}' when
an instance starts.
                This script starts a Python web server defined in the `server.py`
script. The web server
                listens to HTTP requests on port 80 and responds to requests to
'/' and to '/healthcheck'.
                For demo purposes, this server is run as the root user. In
production, the best practice is to
                run a web server, such as Apache, with least-privileged
credentials.

                The template also defines an IAM policy that each instance uses
to assume a role that grants
                permissions to access the DynamoDB recommendation table and
Systems Manager parameters
                that control the flow of the demo.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        templateCreator.createTemplate(policyFile, policyName, profileName,
startScript, templateName, roleName);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println(
                "Creating an EC2 Auto Scaling group that maintains three EC2
instances, each in a different Availability Zone.");
        System.out.println("*** Wait 30 secs for the VPC to be created");
        TimeUnit.SECONDS.sleep(30);
        AutoScaler autoScaler = new AutoScaler();
        String[] zones = autoScaler.createGroup(3, templateName,
autoScalingGroupName);

        System.out.println("""
                At this point, you have EC2 instances created. Once each instance
starts, it listens for
                HTTP requests. You can see these instances in the console or
continue with the demo.
                Press Enter when you're ready to continue.
                """);

        in.nextLine();
        System.out.println(DASHES);
```

```
        System.out.println(DASHES);
        System.out.println("Creating variables that control the flow of the
demo.");
        ParameterHelper paramHelper = new ParameterHelper();
        paramHelper.reset();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("""
                Creating an Elastic Load Balancing target group and load
balancer. The target group
                defines how the load balancer connects to instances. The load
balancer provides a
                single endpoint where clients connect and dispatches requests to
instances in the group.
                """);

        String vpcId = autoScaler.getDefaultVPC();
        List<Subnet> subnets = autoScaler.getSubnets(vpcId, zones);
        System.out.println("You have retrieved a list with " + subnets.size() + "
subnets");
        String targetGroupArn = loadBalancer.createTargetGroup(protocol, port,
vpcId, targetGroupName);
        String elbDnsName = loadBalancer.createLoadBalancer(subnets,
targetGroupArn, lbName, port, protocol);
        autoScaler.attachLoadBalancerTargetGroup(autoScalingGroupName,
targetGroupArn);
        System.out.println("Verifying access to the load balancer endpoint...");
        boolean wasSuccessul =
loadBalancer.verifyLoadBalancerEndpoint(elbDnsName);
        if (!wasSuccessul) {
            System.out.println("Couldn't connect to the load balancer, verifying
that the port is open...");
            CloseableHttpClient httpClient = HttpClients.createDefault();

            // Create an HTTP GET request to "http://checkip.amazonaws.com"
            HttpGet httpGet = new HttpGet("http://checkip.amazonaws.com");
            try {
                // Execute the request and get the response
                HttpResponse response = httpClient.execute(httpGet);

                // Read the response content.
```

```
                String ipAddress =
IOUtils.toString(response.getEntity().getContent(),
StandardCharsets.UTF_8).trim();

                // Print the public IP address.
                System.out.println("Public IP Address: " + ipAddress);
                GroupInfo groupInfo = autoScaler.verifyInboundPort(vpcId, port,
ipAddress);
                if (!groupInfo.isPortOpen()) {
                    System.out.println("""
                            For this example to work, the default security group
for your default VPC must
                            allow access from this computer. You can either add
it automatically from this
                            example or add it yourself using the AWS Management
Console.
                            """);

                    System.out.println(
                            "Do you want to add a rule to security group " +
groupInfo.getGroupName() + " to allow");
                    System.out.println("inbound traffic on port " + port + " from
your computer's IP address (y/n) ");
                    String ans = in.nextLine();
                    if ("y".equalsIgnoreCase(ans)) {
                        autoScaler.openInboundPort(groupInfo.getGroupName(),
String.valueOf(port), ipAddress);
                        System.out.println("Security group rule added.");
                    } else {
                        System.out.println("No security group rule added.");
                    }
                }

            } catch (AutoScalingException e) {
                e.printStackTrace();
            }
        } else if (wasSuccessul) {
            System.out.println("Your load balancer is ready. You can access it by
browsing to:");
            System.out.println("\t http://" + elbDnsName);
        } else {
            System.out.println("Couldn't get a successful response from the load
balancer endpoint. Troubleshoot by");
```

```
            System.out.println("manually verifying that your VPC and security
group are configured correctly and that");
            System.out.println("you can successfully make a GET request to the
load balancer.");
        }

        System.out.println("Press Enter when you're ready to continue with the
demo.");
        in.nextLine();
    }

    // A method that controls the demo part of the Java program.
    public static void demo(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
        ParameterHelper paramHelper = new ParameterHelper();
        System.out.println("Read the ssm_only_policy.json file");
        String ssmOnlyPolicy = readFileAsString(ssmJSON);

        System.out.println("Resetting parameters to starting values for demo.");
        paramHelper.reset();

        System.out.println(
                """
                        This part of the demonstration shows how to toggle
different parts of the system
                        to create situations where the web service fails, and
shows how using a resilient
                        architecture can keep the web service running in spite
of these failures.

                        At the start, the load balancer endpoint returns
recommendations and reports that all targets are healthy.
                       """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        The web service running on the EC2 instances gets
recommendations by querying a DynamoDB table.
                        The table name is contained in a Systems Manager
parameter named self.param_helper.table.
                        To simulate a failure of the recommendation service,
let's set this parameter to name a non-existent table.
                       """);
```

```
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        System.out.println(
                """
                        \nNow, sending a GET request to the load balancer
endpoint returns a failure code. But, the service reports as
                        healthy to the load balancer because shallow health
checks don't check for failure of the recommendation service.
                        """);
        demoChoices(loadBalancer);

        System.out.println(
                """
                        Instead of failing when the recommendation service fails,
the web service can return a static response.
                        While this is not a perfect solution, it presents the
customer with a somewhat better experience than failure.
                        """);
        paramHelper.put(paramHelper.failureResponse, "static");

        System.out.println("""
                Now, sending a GET request to the load balancer endpoint returns
a static response.
                The service still reports as healthy because health checks are
still shallow.
                """);
        demoChoices(loadBalancer);

        System.out.println("Let's reinstate the recommendation service.");
        paramHelper.put(paramHelper.tableName, paramHelper.dyntable);

        System.out.println("""
                Let's also substitute bad credentials for one of the instances in
the target group so that it can't
                access the DynamoDB recommendation table. We will get an instance
id value.
                """);

        LaunchTemplateCreator templateCreator = new LaunchTemplateCreator();
        AutoScaler autoScaler = new AutoScaler();

        // Create a new instance profile based on badCredsProfileName.
        templateCreator.createInstanceProfile(policyFile, policyName,
badCredsProfileName, roleName);
```

```
        String badInstanceId = autoScaler.getBadInstance(autoScalingGroupName);
        System.out.println("The bad instance id values used for this demo is " +
badInstanceId);

        String profileAssociationId =
autoScaler.getInstanceProfile(badInstanceId);
        System.out.println("The association Id value is " +
profileAssociationId);
        System.out.println("Replacing the profile for instance " + badInstanceId
                + " with a profile that contains bad credentials");
        autoScaler.replaceInstanceProfile(badInstanceId, badCredsProfileName,
profileAssociationId);

        System.out.println(
                """
                        Now, sending a GET request to the load balancer endpoint
returns either a recommendation or a static response,
                        depending on which instance is selected by the load
balancer.
                        """);

        demoChoices(loadBalancer);

        System.out.println("""
                Let's implement a deep health check. For this demo, a deep health
check tests whether
                the web service can access the DynamoDB table that it depends on
for recommendations. Note that
                the deep health check is only for ELB routing and not for Auto
Scaling instance health.
                This kind of deep health check is not recommended for Auto
Scaling instance health, because it
                risks accidental termination of all instances in the Auto Scaling
group when a dependent service fails.
                """);

        System.out.println("""
                By implementing deep health checks, the load balancer can detect
when one of the instances is failing
                and take that instance out of rotation.
                """);

        paramHelper.put(paramHelper.healthCheck, "deep");
```

```
        System.out.println("""
                Now, checking target health indicates that the instance with bad
credentials
                is unhealthy. Note that it might take a minute or two for the
load balancer to detect the unhealthy
                instance. Sending a GET request to the load balancer endpoint
always returns a recommendation, because
                the load balancer takes unhealthy instances out of its rotation.
                """);

        demoChoices(loadBalancer);

        System.out.println(
                """
                        Because the instances in this demo are controlled by an
auto scaler, the simplest way to fix an unhealthy
                        instance is to terminate it and let the auto scaler start
a new instance to replace it.
                        """);
        autoScaler.terminateInstance(badInstanceId);

        System.out.println("""
                Even while the instance is terminating and the new instance is
starting, sending a GET
                request to the web service continues to get a successful
recommendation response because
                the load balancer routes requests to the healthy instances. After
the replacement instance
                starts and reports as healthy, it is included in the load
balancing rotation.
                Note that terminating and replacing an instance typically takes
several minutes, during which time you
                can see the changing health check status until the new instance
is running and healthy.
                """);

        demoChoices(loadBalancer);
        System.out.println(
                "If the recommendation service fails now, deep health checks mean
all instances report as unhealthy.");
        paramHelper.put(paramHelper.tableName, "this-is-not-a-table");

        demoChoices(loadBalancer);
        paramHelper.reset();
```

```
   }

   public static void demoChoices(LoadBalancer loadBalancer) throws IOException,
InterruptedException {
       String[] actions = {
               "Send a GET request to the load balancer endpoint.",
               "Check the health of load balancer targets.",
               "Go to the next part of the demo."
       };
       Scanner scanner = new Scanner(System.in);

       while (true) {
           System.out.println("-".repeat(88));
           System.out.println("See the current state of the service by selecting
one of the following choices:");
           for (int i = 0; i < actions.length; i++) {
               System.out.println(i + ": " + actions[i]);
           }

           try {
               System.out.print("\nWhich action would you like to take? ");
               int choice = scanner.nextInt();
               System.out.println("-".repeat(88));

               switch (choice) {
                   case 0 -> {
                       System.out.println("Request:\n");
                       System.out.println("GET http://" +
loadBalancer.getEndpoint(lbName));
                       CloseableHttpClient httpClient =
HttpClients.createDefault();

                       // Create an HTTP GET request to the ELB.
                       HttpGet httpGet = new HttpGet("http://" +
loadBalancer.getEndpoint(lbName));

                       // Execute the request and get the response.
                       HttpResponse response = httpClient.execute(httpGet);
                       int statusCode =
response.getStatusLine().getStatusCode();
                       System.out.println("HTTP Status Code: " + statusCode);

                       // Display the JSON response
                       BufferedReader reader = new BufferedReader(
```

```
                                new
InputStreamReader(response.getEntity().getContent()));
                        StringBuilder jsonResponse = new StringBuilder();
                        String line;
                        while ((line = reader.readLine()) != null) {
                            jsonResponse.append(line);
                        }
                        reader.close();

                        // Print the formatted JSON response.
                        System.out.println("Full Response:\n");
                        System.out.println(jsonResponse.toString());

                        // Close the HTTP client.
                        httpClient.close();

                    }
                    case 1 -> {
                        System.out.println("\nChecking the health of load
balancer targets:\n");
                        List<TargetHealthDescription> health =
loadBalancer.checkTargetHealth(targetGroupName);
                        for (TargetHealthDescription target : health) {
                            System.out.printf("\tTarget %s on port %d is %s%n",
target.target().id(),
                                    target.target().port(),
target.targetHealth().stateAsString());
                        }
                        System.out.println("""
                                Note that it can take a minute or two for the
health check to update
                                after changes are made.
                                """);
                    }
                    case 2 -> {
                        System.out.println("\nOkay, let's move on.");
                        System.out.println("-".repeat(88));
                        return; // Exit the method when choice is 2
                    }
                    default -> System.out.println("You must choose a value
between 0-2. Please select again.");
                }

            } catch (java.util.InputMismatchException e) {
```

```
                System.out.println("Invalid input. Please select again.");
                scanner.nextLine(); // Clear the input buffer.
            }
        }
    }

    public static String readFileAsString(String filePath) throws IOException {
        byte[] bytes = Files.readAllBytes(Paths.get(filePath));
        return new String(bytes);
    }
}
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
public class AutoScaler {

    private static Ec2Client ec2Client;
    private static AutoScalingClient autoScalingClient;
    private static IamClient iamClient;

    private static SsmClient ssmClient;

    private IamClient getIAMClient() {
        if (iamClient == null) {
            iamClient = IamClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return iamClient;
    }

    private SsmClient getSSMClient() {
        if (ssmClient == null) {
            ssmClient = SsmClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ssmClient;
    }

    private Ec2Client getEc2Client() {
        if (ec2Client == null) {
```

```
            ec2Client = Ec2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return ec2Client;
    }

    private AutoScalingClient getAutoScalingClient() {
        if (autoScalingClient == null) {
            autoScalingClient = AutoScalingClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return autoScalingClient;
    }

    /**
     * Terminates and instances in an EC2 Auto Scaling group. After an instance
is
     * terminated, it can no longer be accessed.
     */
    public void terminateInstance(String instanceId) {
        TerminateInstanceInAutoScalingGroupRequest terminateInstanceIRequest =
TerminateInstanceInAutoScalingGroupRequest
                .builder()
                .instanceId(instanceId)
                .shouldDecrementDesiredCapacity(false)
                .build();


getAutoScalingClient().terminateInstanceInAutoScalingGroup(terminateInstanceIRequest);
        System.out.format("Terminated instance %s.", instanceId);
    }

    /**
     * Replaces the profile associated with a running instance. After the profile
is
     * replaced, the instance is rebooted to ensure that it uses the new profile.
     * When
     * the instance is ready, Systems Manager is used to restart the Python web
     * server.
     */
    public void replaceInstanceProfile(String instanceId, String
newInstanceProfileName, String profileAssociationId)
```

```
            throws InterruptedException {
        // Create an IAM instance profile specification.
        software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
iamInstanceProfile =
software.amazon.awssdk.services.ec2.model.IamInstanceProfileSpecification
                .builder()
                .name(newInstanceProfileName) // Make sure
'newInstanceProfileName' is a valid IAM Instance Profile
                                              // name.
                .build();

        // Replace the IAM instance profile association for the EC2 instance.
        ReplaceIamInstanceProfileAssociationRequest replaceRequest =
ReplaceIamInstanceProfileAssociationRequest
                .builder()
                .iamInstanceProfile(iamInstanceProfile)
                .associationId(profileAssociationId) // Make sure
'profileAssociationId' is a valid association ID.
                .build();

        try {
            getEc2Client().replaceIamInstanceProfileAssociation(replaceRequest);
            // Handle the response as needed.
        } catch (Ec2Exception e) {
            // Handle exceptions, log, or report the error.
            System.err.println("Error: " + e.getMessage());
        }
        System.out.format("Replaced instance profile for association %s with
profile %s.", profileAssociationId,
                newInstanceProfileName);
        TimeUnit.SECONDS.sleep(15);
        boolean instReady = false;
        int tries = 0;

        // Reboot after 60 seconds
        while (!instReady) {
            if (tries % 6 == 0) {
                getEc2Client().rebootInstances(RebootInstancesRequest.builder()
                        .instanceIds(instanceId)
                        .build());
                System.out.println("Rebooting instance " + instanceId + " and
waiting for it to be ready.");
            }
            tries++;
```

```
            try {
                TimeUnit.SECONDS.sleep(10);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }

            DescribeInstanceInformationResponse informationResponse =
getSSMClient().describeInstanceInformation();
            List<InstanceInformation> instanceInformationList =
informationResponse.instanceInformationList();
            for (InstanceInformation info : instanceInformationList) {
                if (info.instanceId().equals(instanceId)) {
                    instReady = true;
                    break;
                }
            }
        }

        SendCommandRequest sendCommandRequest = SendCommandRequest.builder()
                .instanceIds(instanceId)
                .documentName("AWS-RunShellScript")
                .parameters(Collections.singletonMap("commands",
                        Collections.singletonList("cd / && sudo python3 server.py
80")))
                .build();

        getSSMClient().sendCommand(sendCommandRequest);
        System.out.println("Restarted the Python web server on instance " +
instanceId + ".");
    }

    public void openInboundPort(String secGroupId, String port, String ipAddress)
{
        AuthorizeSecurityGroupIngressRequest ingressRequest =
AuthorizeSecurityGroupIngressRequest.builder()
                .groupName(secGroupId)
                .cidrIp(ipAddress)
                .fromPort(Integer.parseInt(port))
                .build();

        getEc2Client().authorizeSecurityGroupIngress(ingressRequest);
        System.out.format("Authorized ingress to %s on port %s from %s.",
secGroupId, port, ipAddress);
    }
```

```
   /**
    * Detaches a role from an instance profile, detaches policies from the role,
    * and deletes all the resources.
    */
   public void deleteInstanceProfile(String roleName, String profileName) {
       try {
           software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
getInstanceProfileRequest =
software.amazon.awssdk.services.iam.model.GetInstanceProfileRequest
                   .builder()
                   .instanceProfileName(profileName)
                   .build();

           GetInstanceProfileResponse response =
getIAMClient().getInstanceProfile(getInstanceProfileRequest);
           String name = response.instanceProfile().instanceProfileName();
           System.out.println(name);

           RemoveRoleFromInstanceProfileRequest profileRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .roleName(roleName)
                   .build();

           getIAMClient().removeRoleFromInstanceProfile(profileRequest);
           DeleteInstanceProfileRequest deleteInstanceProfileRequest =
DeleteInstanceProfileRequest.builder()
                   .instanceProfileName(profileName)
                   .build();

           getIAMClient().deleteInstanceProfile(deleteInstanceProfileRequest);
           System.out.println("Deleted instance profile " + profileName);

           DeleteRoleRequest deleteRoleRequest = DeleteRoleRequest.builder()
                   .roleName(roleName)
                   .build();

           // List attached role policies.
           ListAttachedRolePoliciesResponse rolesResponse = getIAMClient()
                   .listAttachedRolePolicies(role -> role.roleName(roleName));
           List<AttachedPolicy> attachedPolicies =
rolesResponse.attachedPolicies();
           for (AttachedPolicy attachedPolicy : attachedPolicies) {
```

```
                DetachRolePolicyRequest request =
DetachRolePolicyRequest.builder()
                        .roleName(roleName)
                        .policyArn(attachedPolicy.policyArn())
                        .build();

                getIAMClient().detachRolePolicy(request);
                System.out.println("Detached and deleted policy " +
attachedPolicy.policyName());
            }

            getIAMClient().deleteRole(deleteRoleRequest);
            System.out.println("Instance profile and role deleted.");

        } catch (IamException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }

    public void deleteTemplate(String templateName) {
        getEc2Client().deleteLaunchTemplate(name ->
name.launchTemplateName(templateName));
        System.out.format(templateName + " was deleted.");
    }

    public void deleteAutoScaleGroup(String groupName) {
        DeleteAutoScalingGroupRequest deleteAutoScalingGroupRequest =
DeleteAutoScalingGroupRequest.builder()
                .autoScalingGroupName(groupName)
                .forceDelete(true)
                .build();


getAutoScalingClient().deleteAutoScalingGroup(deleteAutoScalingGroupRequest);
        System.out.println(groupName + " was deleted.");
    }

    /*
     * Verify the default security group of the specified VPC allows ingress from
     * this
     * computer. This can be done by allowing ingress from this computer's IP
     * address. In some situations, such as connecting from a corporate network,
you
```

```
    * must instead specify a prefix list ID. You can also temporarily open the
port
    * to
    * any IP address while running this example. If you do, be sure to remove
    * public
    * access when you're done.
    *
    */
   public GroupInfo verifyInboundPort(String VPC, int port, String ipAddress) {
       boolean portIsOpen = false;
       GroupInfo groupInfo = new GroupInfo();
       try {
           Filter filter = Filter.builder()
                   .name("group-name")
                   .values("default")
                   .build();

           Filter filter1 = Filter.builder()
                   .name("vpc-id")
                   .values(VPC)
                   .build();

           DescribeSecurityGroupsRequest securityGroupsRequest =
DescribeSecurityGroupsRequest.builder()
                   .filters(filter, filter1)
                   .build();

           DescribeSecurityGroupsResponse securityGroupsResponse =
getEc2Client()
                   .describeSecurityGroups(securityGroupsRequest);
           String securityGroup =
securityGroupsResponse.securityGroups().get(0).groupName();
           groupInfo.setGroupName(securityGroup);

           for (SecurityGroup secGroup :
securityGroupsResponse.securityGroups()) {
               System.out.println("Found security group: " +
secGroup.groupId());

               for (IpPermission ipPermission : secGroup.ipPermissions()) {
                   if (ipPermission.fromPort() == port) {
                       System.out.println("Found inbound rule: " +
ipPermission);
                       for (IpRange ipRange : ipPermission.ipRanges()) {
```

```
                            String cidrIp = ipRange.cidrIp();
                            if (cidrIp.startsWith(ipAddress) ||
cidrIp.equals("0.0.0.0/0")) {
                                System.out.println(cidrIp + " is applicable");
                                portIsOpen = true;
                            }
                        }

                        if (!ipPermission.prefixListIds().isEmpty()) {
                            System.out.println("Prefix lList is applicable");
                            portIsOpen = true;
                        }

                        if (!portIsOpen) {
                            System.out
                                    .println("The inbound rule does not appear to
be open to either this computer's IP,"
                                            + " all IP addresses (0.0.0.0/0), or
to a prefix list ID.");
                        } else {
                            break;
                        }
                    }
                }
            }

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }

        groupInfo.setPortOpen(portIsOpen);
        return groupInfo;
    }

    /*
     * Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
     * Scaling group.
     * The target group specifies how the load balancer forward requests to the
     * instances
     * in the group.
     */
    public void attachLoadBalancerTargetGroup(String asGroupName, String
targetGroupARN) {
        try {
```

```
            AttachLoadBalancerTargetGroupsRequest targetGroupsRequest =
AttachLoadBalancerTargetGroupsRequest.builder()
                    .autoScalingGroupName(asGroupName)
                    .targetGroupARNs(targetGroupARN)
                    .build();


getAutoScalingClient().attachLoadBalancerTargetGroups(targetGroupsRequest);
            System.out.println("Attached load balancer to " + asGroupName);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Creates an EC2 Auto Scaling group with the specified size.
    public String[] createGroup(int groupSize, String templateName, String
autoScalingGroupName) {

        // Get availability zones.

software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
zonesRequest =
software.amazon.awssdk.services.ec2.model.DescribeAvailabilityZonesRequest
                .builder()
                .build();

        DescribeAvailabilityZonesResponse zonesResponse =
getEc2Client().describeAvailabilityZones(zonesRequest);
        List<String> availabilityZoneNames =
zonesResponse.availabilityZones().stream()

.map(software.amazon.awssdk.services.ec2.model.AvailabilityZone::zoneName)
                .collect(Collectors.toList());

        String availabilityZones = String.join(",", availabilityZoneNames);
        LaunchTemplateSpecification specification =
LaunchTemplateSpecification.builder()
                .launchTemplateName(templateName)
                .version("$Default")
                .build();

        String[] zones = availabilityZones.split(",");
```

```
        CreateAutoScalingGroupRequest groupRequest =
CreateAutoScalingGroupRequest.builder()
                .launchTemplate(specification)
                .availabilityZones(zones)
                .maxSize(groupSize)
                .minSize(groupSize)
                .autoScalingGroupName(autoScalingGroupName)
                .build();

        try {
            getAutoScalingClient().createAutoScalingGroup(groupRequest);

        } catch (AutoScalingException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        System.out.println("Created an EC2 Auto Scaling group named " +
autoScalingGroupName);
        return zones;
    }

    public String getDefaultVPC() {
        // Define the filter.
        Filter defaultFilter = Filter.builder()
                .name("is-default")
                .values("true")
                .build();

        software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest request =
software.amazon.awssdk.services.ec2.model.DescribeVpcsRequest
                .builder()
                .filters(defaultFilter)
                .build();

        DescribeVpcsResponse response = getEc2Client().describeVpcs(request);
        return response.vpcs().get(0).vpcId();
    }

    // Gets the default subnets in a VPC for a specified list of Availability
Zones.
    public List<Subnet> getSubnets(String vpcId, String[] availabilityZones) {
        List<Subnet> subnets = null;
        Filter vpcFilter = Filter.builder()
                .name("vpc-id")
```

```
                .values(vpcId)
                .build();

        Filter azFilter = Filter.builder()
                .name("availability-zone")
                .values(availabilityZones)
                .build();

        Filter defaultForAZ = Filter.builder()
                .name("default-for-az")
                .values("true")
                .build();

        DescribeSubnetsRequest request = DescribeSubnetsRequest.builder()
                .filters(vpcFilter, azFilter, defaultForAZ)
                .build();

        DescribeSubnetsResponse response =
getEc2Client().describeSubnets(request);
        subnets = response.subnets();
        return subnets;
    }

    // Gets data about the instances in the EC2 Auto Scaling group.
    public String getBadInstance(String groupName) {
        DescribeAutoScalingGroupsRequest request =
DescribeAutoScalingGroupsRequest.builder()
                .autoScalingGroupNames(groupName)
                .build();

        DescribeAutoScalingGroupsResponse response =
getAutoScalingClient().describeAutoScalingGroups(request);
        AutoScalingGroup autoScalingGroup = response.autoScalingGroups().get(0);
        List<String> instanceIds = autoScalingGroup.instances().stream()
                .map(instance -> instance.instanceId())
                .collect(Collectors.toList());

        String[] instanceIdArray = instanceIds.toArray(new String[0]);
        for (String instanceId : instanceIdArray) {
            System.out.println("Instance ID: " + instanceId);
            return instanceId;
        }
        return "";
    }
```

```
   // Gets data about the profile associated with an instance.
   public String getInstanceProfile(String instanceId) {
       Filter filter = Filter.builder()
               .name("instance-id")
               .values(instanceId)
               .build();

       DescribeIamInstanceProfileAssociationsRequest associationsRequest =
 DescribeIamInstanceProfileAssociationsRequest
               .builder()
               .filters(filter)
               .build();

       DescribeIamInstanceProfileAssociationsResponse response = getEc2Client()
               .describeIamInstanceProfileAssociations(associationsRequest);
       return response.iamInstanceProfileAssociations().get(0).associationId();
   }

   public void deleteRolesPolicies(String policyName, String roleName, String
 InstanceProfile) {
       ListPoliciesRequest listPoliciesRequest =
 ListPoliciesRequest.builder().build();
       ListPoliciesResponse listPoliciesResponse =
 getIAMClient().listPolicies(listPoliciesRequest);
       for (Policy policy : listPoliciesResponse.policies()) {
           if (policy.policyName().equals(policyName)) {
               // List the entities (users, groups, roles) that are attached to
 the policy.

 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
 listEntitiesRequest =
 software.amazon.awssdk.services.iam.model.ListEntitiesForPolicyRequest
                       .builder()
                       .policyArn(policy.arn())
                       .build();
               ListEntitiesForPolicyResponse listEntitiesResponse = iamClient
                       .listEntitiesForPolicy(listEntitiesRequest);
               if (!listEntitiesResponse.policyGroups().isEmpty() || !
listEntitiesResponse.policyUsers().isEmpty()
                       || !listEntitiesResponse.policyRoles().isEmpty()) {
                   // Detach the policy from any entities it is attached to.
                   DetachRolePolicyRequest detachPolicyRequest =
 DetachRolePolicyRequest.builder()
```

```
                            .policyArn(policy.arn())
                            .roleName(roleName) // Specify the name of the IAM
role
                            .build();

                    getIAMClient().detachRolePolicy(detachPolicyRequest);
                    System.out.println("Policy detached from entities.");
                }

                // Now, you can delete the policy.
                DeletePolicyRequest deletePolicyRequest =
DeletePolicyRequest.builder()
                        .policyArn(policy.arn())
                        .build();

                getIAMClient().deletePolicy(deletePolicyRequest);
                System.out.println("Policy deleted successfully.");
                break;
            }
        }

        // List the roles associated with the instance profile
        ListInstanceProfilesForRoleRequest listRolesRequest =
ListInstanceProfilesForRoleRequest.builder()
                .roleName(roleName)
                .build();

        // Detach the roles from the instance profile
        ListInstanceProfilesForRoleResponse listRolesResponse =
iamClient.listInstanceProfilesForRole(listRolesRequest);
        for (software.amazon.awssdk.services.iam.model.InstanceProfile profile :
listRolesResponse.instanceProfiles()) {
            RemoveRoleFromInstanceProfileRequest removeRoleRequest =
RemoveRoleFromInstanceProfileRequest.builder()
                    .instanceProfileName(InstanceProfile)
                    .roleName(roleName) // Remove the extra dot here
                    .build();

            getIAMClient().removeRoleFromInstanceProfile(removeRoleRequest);
            System.out.println("Role " + roleName + " removed from instance
profile " + InstanceProfile);
        }

        // Delete the instance profile after removing all roles
```

```
        DeleteInstanceProfileRequest deleteInstanceProfileRequest =
 DeleteInstanceProfileRequest.builder()
                .instanceProfileName(InstanceProfile)
                .build();

        getIAMClient().deleteInstanceProfile(r ->
 r.instanceProfileName(InstanceProfile));
        System.out.println(InstanceProfile + " Deleted");
        System.out.println("All roles and policies are deleted.");
    }
}
```

创建一个包含弹性负载均衡操作的类。

```
public class LoadBalancer {
    public ElasticLoadBalancingV2Client elasticLoadBalancingV2Client;

    public ElasticLoadBalancingV2Client getLoadBalancerClient() {
        if (elasticLoadBalancingV2Client == null) {
            elasticLoadBalancingV2Client = ElasticLoadBalancingV2Client.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }

        return elasticLoadBalancingV2Client;
    }

    // Checks the health of the instances in the target group.
    public List<TargetHealthDescription> checkTargetHealth(String
 targetGroupName) {
        DescribeTargetGroupsRequest targetGroupsRequest =
 DescribeTargetGroupsRequest.builder()
                .names(targetGroupName)
                .build();

        DescribeTargetGroupsResponse tgResponse =
 getLoadBalancerClient().describeTargetGroups(targetGroupsRequest);

        DescribeTargetHealthRequest healthRequest =
 DescribeTargetHealthRequest.builder()

 .targetGroupArn(tgResponse.targetGroups().get(0).targetGroupArn())
```

```
                .build();

        DescribeTargetHealthResponse healthResponse =
getLoadBalancerClient().describeTargetHealth(healthRequest);
        return healthResponse.targetHealthDescriptions();
    }

    // Gets the HTTP endpoint of the load balancer.
    public String getEndpoint(String lbName) {
        DescribeLoadBalancersResponse res = getLoadBalancerClient()
                .describeLoadBalancers(describe -> describe.names(lbName));
        return res.loadBalancers().get(0).dnsName();
    }

    // Deletes a load balancer.
    public void deleteLoadBalancer(String lbName) {
        try {
            // Use a waiter to delete the Load Balancer.
            DescribeLoadBalancersResponse res = getLoadBalancerClient()
                    .describeLoadBalancers(describe -> describe.names(lbName));
            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()

.loadBalancerArns(res.loadBalancers().get(0).loadBalancerArn())
                    .build();

            getLoadBalancerClient().deleteLoadBalancer(
                    builder ->
builder.loadBalancerArn(res.loadBalancers().get(0).loadBalancerArn()));
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancersDeleted(request);
            waiterResponse.matched().response().ifPresent(System.out::println);

        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(lbName + " was deleted.");
    }

    // Deletes the target group.
    public void deleteTargetGroup(String targetGroupName) {
```

```
        try {
            DescribeTargetGroupsResponse res = getLoadBalancerClient()
                    .describeTargetGroups(describe ->
describe.names(targetGroupName));
            getLoadBalancerClient()
                    .deleteTargetGroup(builder ->
builder.targetGroupArn(res.targetGroups().get(0).targetGroupArn()));
        } catch (ElasticLoadBalancingV2Exception e) {
            System.err.println(e.awsErrorDetails().errorMessage());
        }
        System.out.println(targetGroupName + " was deleted.");
    }

    // Verify this computer can successfully send a GET request to the load
balancer
    // endpoint.
    public boolean verifyLoadBalancerEndpoint(String elbDnsName) throws
IOException, InterruptedException {
        boolean success = false;
        int retries = 3;
        CloseableHttpClient httpClient = HttpClients.createDefault();

        // Create an HTTP GET request to the ELB.
        HttpGet httpGet = new HttpGet("http://" + elbDnsName);
        try {
            while ((!success) && (retries > 0)) {
                // Execute the request and get the response.
                HttpResponse response = httpClient.execute(httpGet);
                int statusCode = response.getStatusLine().getStatusCode();
                System.out.println("HTTP Status Code: " + statusCode);
                if (statusCode == 200) {
                    success = true;
                } else {
                    retries--;
                    System.out.println("Got connection error from load balancer
endpoint, retrying...");
                    TimeUnit.SECONDS.sleep(15);
                }
            }

        } catch (org.apache.http.conn.HttpHostConnectException e) {
            System.out.println(e.getMessage());
        }
```

```
        System.out.println("Status.." + success);
        return success;
    }

    /*
     * Creates an Elastic Load Balancing target group. The target group specifies
     * how
     * the load balancer forward requests to instances in the group and how
instance
     * health is checked.
     */
    public String createTargetGroup(String protocol, int port, String vpcId,
String targetGroupName) {
        CreateTargetGroupRequest targetGroupRequest =
CreateTargetGroupRequest.builder()
                .healthCheckPath("/healthcheck")
                .healthCheckTimeoutSeconds(5)
                .port(port)
                .vpcId(vpcId)
                .name(targetGroupName)
                .protocol(protocol)
                .build();

        CreateTargetGroupResponse targetGroupResponse =
getLoadBalancerClient().createTargetGroup(targetGroupRequest);
        String targetGroupArn =
targetGroupResponse.targetGroups().get(0).targetGroupArn();
        String targetGroup =
targetGroupResponse.targetGroups().get(0).targetGroupName();
        System.out.println("The " + targetGroup + " was created with ARN" +
targetGroupArn);
        return targetGroupArn;
    }

    /*
     * Creates an Elastic Load Balancing load balancer that uses the specified
     * subnets
     * and forwards requests to the specified target group.
     */
    public String createLoadBalancer(List<Subnet> subnetIds, String
targetGroupARN, String lbName, int port,
            String protocol) {
        try {
            List<String> subnetIdStrings = subnetIds.stream()
```

```
                    .map(Subnet::subnetId)
                    .collect(Collectors.toList());

            CreateLoadBalancerRequest balancerRequest =
CreateLoadBalancerRequest.builder()
                    .subnets(subnetIdStrings)
                    .name(lbName)
                    .scheme("internet-facing")
                    .build();

            // Create and wait for the load balancer to become available.
            CreateLoadBalancerResponse lsResponse =
getLoadBalancerClient().createLoadBalancer(balancerRequest);
            String lbARN = lsResponse.loadBalancers().get(0).loadBalancerArn();

            ElasticLoadBalancingV2Waiter loadBalancerWaiter =
getLoadBalancerClient().waiter();
            DescribeLoadBalancersRequest request =
DescribeLoadBalancersRequest.builder()
                    .loadBalancerArns(lbARN)
                    .build();

            System.out.println("Waiting for Load Balancer " + lbName + " to
become available.");
            WaiterResponse<DescribeLoadBalancersResponse> waiterResponse =
loadBalancerWaiter
                    .waitUntilLoadBalancerAvailable(request);
            waiterResponse.matched().response().ifPresent(System.out::println);
            System.out.println("Load Balancer " + lbName + " is available.");

            // Get the DNS name (endpoint) of the load balancer.
            String lbDNSName = lsResponse.loadBalancers().get(0).dnsName();
            System.out.println("*** Load Balancer DNS Name: " + lbDNSName);

            // Create a listener for the load balance.
            Action action = Action.builder()
                    .targetGroupArn(targetGroupARN)
                    .type("forward")
                    .build();

            CreateListenerRequest listenerRequest =
CreateListenerRequest.builder()

.loadBalancerArn(lsResponse.loadBalancers().get(0).loadBalancerArn())
```

```
                    .defaultActions(action)
                    .port(port)
                    .protocol(protocol)
                    .build();

            getLoadBalancerClient().createListener(listenerRequest);
            System.out.println("Created listener to forward traffic from load
 balancer " + lbName + " to target group "
                    + targetGroupARN);

            // Return the load balancer DNS name.
            return lbDNSName;

        } catch (ElasticLoadBalancingV2Exception e) {
            e.printStackTrace();
        }
        return "";
    }
}
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
public class Database {

    private static DynamoDbClient dynamoDbClient;

    public static DynamoDbClient getDynamoDbClient() {
        if (dynamoDbClient == null) {
            dynamoDbClient = DynamoDbClient.builder()
                    .region(Region.US_EAST_1)
                    .build();
        }
        return dynamoDbClient;
    }

    // Checks to see if the Amazon DynamoDB table exists.
    private boolean doesTableExist(String tableName) {
        try {
            // Describe the table and catch any exceptions.
            DescribeTableRequest describeTableRequest =
 DescribeTableRequest.builder()
                    .tableName(tableName)
```

```
                    .build();

            getDynamoDbClient().describeTable(describeTableRequest);
            System.out.println("Table '" + tableName + "' exists.");
            return true;

        } catch (ResourceNotFoundException e) {
            System.out.println("Table '" + tableName + "' does not exist.");
        } catch (DynamoDbException e) {
            System.err.println("Error checking table existence: " +
e.getMessage());
        }
        return false;
    }

    /*
     * Creates a DynamoDB table to use a recommendation service. The table has a
     * hash key named 'MediaType' that defines the type of media recommended,
such
     * as
     * Book or Movie, and a range key named 'ItemId' that, combined with the
     * MediaType,
     * forms a unique identifier for the recommended item.
     */
    public void createTable(String tableName, String fileName) throws IOException
{
        // First check to see if the table exists.
        boolean doesExist = doesTableExist(tableName);
        if (!doesExist) {
            DynamoDbWaiter dbWaiter = getDynamoDbClient().waiter();
            CreateTableRequest createTableRequest = CreateTableRequest.builder()
                    .tableName(tableName)
                    .attributeDefinitions(
                            AttributeDefinition.builder()
                                    .attributeName("MediaType")
                                    .attributeType(ScalarAttributeType.S)
                                    .build(),
                            AttributeDefinition.builder()
                                    .attributeName("ItemId")
                                    .attributeType(ScalarAttributeType.N)
                                    .build())
                    .keySchema(
                            KeySchemaElement.builder()
                                    .attributeName("MediaType")
```

```
                                .keyType(KeyType.HASH)
                                .build(),
                        KeySchemaElement.builder()
                                .attributeName("ItemId")
                                .keyType(KeyType.RANGE)
                                .build())
                .provisionedThroughput(
                        ProvisionedThroughput.builder()
                                .readCapacityUnits(5L)
                                .writeCapacityUnits(5L)
                                .build())
                .build();

        getDynamoDbClient().createTable(createTableRequest);
        System.out.println("Creating table " + tableName + "...");

        // Wait until the Amazon DynamoDB table is created.
        DescribeTableRequest tableRequest = DescribeTableRequest.builder()
                .tableName(tableName)
                .build();

        WaiterResponse<DescribeTableResponse> waiterResponse =
dbWaiter.waitUntilTableExists(tableRequest);
        waiterResponse.matched().response().ifPresent(System.out::println);
        System.out.println("Table " + tableName + " created.");

        // Add records to the table.
        populateTable(fileName, tableName);
    }
}

public void deleteTable(String tableName) {
    getDynamoDbClient().deleteTable(table -> table.tableName(tableName));
    System.out.println("Table " + tableName + " deleted.");
}

// Populates the table with data located in a JSON file using the DynamoDB
// enhanced client.
public void populateTable(String fileName, String tableName) throws
IOException {
    DynamoDbEnhancedClient enhancedClient = DynamoDbEnhancedClient.builder()
            .dynamoDbClient(getDynamoDbClient())
            .build();
    ObjectMapper objectMapper = new ObjectMapper();
```

```
        File jsonFile = new File(fileName);
        JsonNode rootNode = objectMapper.readTree(jsonFile);

        DynamoDbTable<Recommendation> mappedTable =
 enhancedClient.table(tableName,
                TableSchema.fromBean(Recommendation.class));
        for (JsonNode currentNode : rootNode) {
            String mediaType = currentNode.path("MediaType").path("S").asText();
            int itemId = currentNode.path("ItemId").path("N").asInt();
            String title = currentNode.path("Title").path("S").asText();
            String creator = currentNode.path("Creator").path("S").asText();

            // Create a Recommendation object and set its properties.
            Recommendation rec = new Recommendation();
            rec.setMediaType(mediaType);
            rec.setItemId(itemId);
            rec.setTitle(title);
            rec.setCreator(creator);

            // Put the item into the DynamoDB table.
            mappedTable.putItem(rec); // Add the Recommendation to the list.
        }
        System.out.println("Added all records to the " + tableName);
    }
}
```

创建一个包含 Systems Manager 操作的类。

```
public class ParameterHelper {

    String tableName = "doc-example-resilient-architecture-table";
    String dyntable = "doc-example-recommendation-service";
    String failureResponse = "doc-example-resilient-architecture-failure-
response";
    String healthCheck = "doc-example-resilient-architecture-health-check";

    public void reset() {
        put(dyntable, tableName);
        put(failureResponse, "none");
        put(healthCheck, "shallow");
    }
```

```
    public void put(String name, String value) {
        SsmClient ssmClient = SsmClient.builder()
                .region(Region.US_EAST_1)
                .build();

        PutParameterRequest parameterRequest = PutParameterRequest.builder()
                .name(name)
                .value(value)
                .overwrite(true)
                .type("String")
                .build();

        ssmClient.putParameter(parameterRequest);
        System.out.printf("Setting demo parameter %s to '%s'.", name, value);
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers

- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
#!/usr/bin/env node
// Copyright Amazon.com, Inc. or its affiliates. All Rights Reserved.
// SPDX-License-Identifier: Apache-2.0

import {
  Scenario,
  parseScenarioArgs,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";

/**
 * The workflow steps are split into three stages:
 *   - deploy
 *   - demo
 *   - destroy
 *
 * Each of these stages has a corresponding file prefixed with steps-*.
 */
import { deploySteps } from "./steps-deploy.js";
```

```
import { demoSteps } from "./steps-demo.js";
import { destroySteps } from "./steps-destroy.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
export const scenarios = {
  // Deploys all resources necessary for the workflow.
  deploy: new Scenario("Resilient Workflow - Deploy", deploySteps, context),
  // Demonstrates how a fragile web service can be made more resilient.
  demo: new Scenario("Resilient Workflow - Demo", demoSteps, context),
  // Destroys the resources created for the workflow.
  destroy: new Scenario("Resilient Workflow - Destroy", destroySteps, context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Resilient Workflow",
    synopsis:
      "node index.js --scenario <deploy | demo | destroy> [-h|--help] [-y|--yes]
 [-v|--verbose]",
    description: "Deploy and interact with scalable EC2 instances.",
  });
}
```

创建部署所有资源的步骤。

```
import { join } from "node:path";
import { readFileSync, writeFileSync } from "node:fs";
import axios from "axios";
```

```
import {
  BatchWriteItemCommand,
  CreateTableCommand,
  DynamoDBClient,
  waitUntilTableExists,
} from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  CreateKeyPairCommand,
  CreateLaunchTemplateCommand,
  DescribeAvailabilityZonesCommand,
  DescribeVpcsCommand,
  DescribeSubnetsCommand,
  DescribeSecurityGroupsCommand,
  AuthorizeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  AttachRolePolicyCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import { SSMClient, GetParameterCommand } from "@aws-sdk/client-ssm";
import {
  CreateAutoScalingGroupCommand,
  AutoScalingClient,
  AttachLoadBalancerTargetGroupsCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  CreateListenerCommand,
  CreateLoadBalancerCommand,
  CreateTargetGroupCommand,
  ElasticLoadBalancingV2Client,
  waitUntilLoadBalancerAvailable,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
```

```
import { saveState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH, ROOT } from "./constants.js";
import { initParamsSteps } from "./steps-reset-params.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const deploySteps = [
  new ScenarioOutput("introduction", MESSAGES.introduction, { header: true }),
  new ScenarioInput("confirmDeployment", MESSAGES.confirmDeployment, {
    type: "confirm",
  }),
  new ScenarioAction(
    "handleConfirmDeployment",
    (c) => c.confirmDeployment === false && process.exit(),
  ),
  new ScenarioOutput(
    "creatingTable",
    MESSAGES.creatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("createTable", async () => {
    const client = new DynamoDBClient({});
    await client.send(
      new CreateTableCommand({
        TableName: NAMES.tableName,
        ProvisionedThroughput: {
          ReadCapacityUnits: 5,
          WriteCapacityUnits: 5,
        },
        AttributeDefinitions: [
          {
            AttributeName: "MediaType",
            AttributeType: "S",
          },
          {
            AttributeName: "ItemId",
            AttributeType: "N",
          },
        ],
        KeySchema: [
          {
            AttributeName: "MediaType",
```

```
            KeyType: "HASH",
          },
          {
            AttributeName: "ItemId",
            KeyType: "RANGE",
          },
        ],
      }),
    );
    await waitUntilTableExists({ client }, { TableName: NAMES.tableName });
  }),
  new ScenarioOutput(
    "createdTable",
    MESSAGES.createdTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
    "populatingTable",
    MESSAGES.populatingTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioAction("populateTable", () => {
    const client = new DynamoDBClient({});
    /**
     * @type {{ default: import("@aws-sdk/client-dynamodb").PutRequest['Item']
[] }}
     */
    const recommendations = JSON.parse(
      readFileSync(join(RESOURCES_PATH, "recommendations.json")),
    );

    return client.send(
      new BatchWriteItemCommand({
        RequestItems: {
          [NAMES.tableName]: recommendations.map((item) => ({
            PutRequest: { Item: item },
          })),
        },
      }),
    );
  }),
  new ScenarioOutput(
    "populatedTable",
    MESSAGES.populatedTable.replace("${TABLE_NAME}", NAMES.tableName),
  ),
  new ScenarioOutput(
```

```
    "creatingKeyPair",
    MESSAGES.creatingKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioAction("createKeyPair", async () => {
    const client = new EC2Client({});
    const { KeyMaterial } = await client.send(
      new CreateKeyPairCommand({
        KeyName: NAMES.keyPairName,
      }),
    );

    writeFileSync(`${NAMES.keyPairName}.pem`, KeyMaterial, { mode: 0o600 });
  }),
  new ScenarioOutput(
    "createdKeyPair",
    MESSAGES.createdKeyPair.replace("${KEY_PAIR_NAME}", NAMES.keyPairName),
  ),
  new ScenarioOutput(
    "creatingInstancePolicy",
    MESSAGES.creatingInstancePolicy.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    ),
  ),
  new ScenarioAction("createInstancePolicy", async (state) => {
    const client = new IAMClient({});
    const {
      Policy: { Arn },
    } = await client.send(
      new CreatePolicyCommand({
        PolicyName: NAMES.instancePolicyName,
        PolicyDocument: readFileSync(
          join(RESOURCES_PATH, "instance_policy.json"),
        ),
      }),
    );
    state.instancePolicyArn = Arn;
  }),
  new ScenarioOutput("createdInstancePolicy", (state) =>
    MESSAGES.createdInstancePolicy
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_POLICY_ARN}", state.instancePolicyArn),
  ),
  new ScenarioOutput(
```

```
    "creatingInstanceRole",
    MESSAGES.creatingInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioAction("createInstanceRole", () => {
    const client = new IAMClient({});
    return client.send(
      new CreateRoleCommand({
        RoleName: NAMES.instanceRoleName,
        AssumeRolePolicyDocument: readFileSync(
          join(ROOT, "assume-role-policy.json"),
        ),
      }),
    );
  }),
  new ScenarioOutput(
    "createdInstanceRole",
    MESSAGES.createdInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    ),
  ),
  new ScenarioOutput(
    "attachingPolicyToRole",
    MESSAGES.attachingPolicyToRole
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName)
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName),
  ),
  new ScenarioAction("attachPolicyToRole", async (state) => {
    const client = new IAMClient({});
    await client.send(
      new AttachRolePolicyCommand({
        RoleName: NAMES.instanceRoleName,
        PolicyArn: state.instancePolicyArn,
      }),
    );
  }),
  new ScenarioOutput(
    "attachedPolicyToRole",
    MESSAGES.attachedPolicyToRole
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
```

```
  ),
  new ScenarioOutput(
    "creatingInstanceProfile",
    MESSAGES.creatingInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    ),
  ),
  new ScenarioAction("createInstanceProfile", async (state) => {
    const client = new IAMClient({});
    const {
      InstanceProfile: { Arn },
    } = await client.send(
      new CreateInstanceProfileCommand({
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
    state.instanceProfileArn = Arn;

    await waitUntilInstanceProfileExists(
      { client },
      { InstanceProfileName: NAMES.instanceProfileName },
    );
  }),
  new ScenarioOutput("createdInstanceProfile", (state) =>
    MESSAGES.createdInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_PROFILE_ARN}", state.instanceProfileArn),
  ),
  new ScenarioOutput(
    "addingRoleToInstanceProfile",
    MESSAGES.addingRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  new ScenarioAction("addRoleToInstanceProfile", () => {
    const client = new IAMClient({});
    return client.send(
      new AddRoleToInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  }),
```

```
  new ScenarioOutput(
    "addedRoleToInstanceProfile",
    MESSAGES.addedRoleToInstanceProfile
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName),
  ),
  ...initParamsSteps,
  new ScenarioOutput("creatingLaunchTemplate", MESSAGES.creatingLaunchTemplate),
  new ScenarioAction("createLaunchTemplate", async () => {
    const ssmClient = new SSMClient({});
    const { Parameter } = await ssmClient.send(
      new GetParameterCommand({
        Name: "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
      }),
    );
    const ec2Client = new EC2Client({});
    await ec2Client.send(
      new CreateLaunchTemplateCommand({
        LaunchTemplateName: NAMES.launchTemplateName,
        LaunchTemplateData: {
          InstanceType: "t3.micro",
          ImageId: Parameter.Value,
          IamInstanceProfile: { Name: NAMES.instanceProfileName },
          UserData: readFileSync(
            join(RESOURCES_PATH, "server_startup_script.sh"),
          ).toString("base64"),
          KeyName: NAMES.keyPairName,
        },
      }),
    );
  }),
  new ScenarioOutput(
    "createdLaunchTemplate",
    MESSAGES.createdLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    ),
  ),
  new ScenarioOutput(
    "creatingAutoScalingGroup",
    MESSAGES.creatingAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    ),
```

```
  ),
  new ScenarioAction("createAutoScalingGroup", async (state) => {
    const ec2Client = new EC2Client({});
    const { AvailabilityZones } = await ec2Client.send(
      new DescribeAvailabilityZonesCommand({}),
    );
    state.availabilityZoneNames = AvailabilityZones.map((az) => az.ZoneName);
    const autoScalingClient = new AutoScalingClient({});
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new CreateAutoScalingGroupCommand({
          AvailabilityZones: state.availabilityZoneNames,
          AutoScalingGroupName: NAMES.autoScalingGroupName,
          LaunchTemplate: {
            LaunchTemplateName: NAMES.launchTemplateName,
            Version: "$Default",
          },
          MinSize: 3,
          MaxSize: 3,
        }),
      ),
    );
  }),
  new ScenarioOutput(
    "createdAutoScalingGroup",
    /**
     * @param {{ availabilityZoneNames: string[] }} state
     */
    (state) =>
      MESSAGES.createdAutoScalingGroup
        .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName)
        .replace(
          "${AVAILABILITY_ZONE_NAMES}",
          state.availabilityZoneNames.join(", "),
        ),
  ),
  new ScenarioInput("confirmContinue", MESSAGES.confirmContinue, {
    type: "confirm",
  }),
  new ScenarioOutput("loadBalancer", MESSAGES.loadBalancer),
  new ScenarioOutput("gettingVpc", MESSAGES.gettingVpc),
  new ScenarioAction("getVpc", async (state) => {
    const client = new EC2Client({});
    const { Vpcs } = await client.send(
```

```
      new DescribeVpcsCommand({
        Filters: [{ Name: "is-default", Values: ["true"] }],
      }),
    );
    state.defaultVpc = Vpcs[0].VpcId;
  }),
  new ScenarioOutput("gotVpc", (state) =>
    MESSAGES.gotVpc.replace("${VPC_ID}", state.defaultVpc),
  ),
  new ScenarioOutput("gettingSubnets", MESSAGES.gettingSubnets),
  new ScenarioAction("getSubnets", async (state) => {
    const client = new EC2Client({});
    const { Subnets } = await client.send(
      new DescribeSubnetsCommand({
        Filters: [
          { Name: "vpc-id", Values: [state.defaultVpc] },
          { Name: "availability-zone", Values: state.availabilityZoneNames },
          { Name: "default-for-az", Values: ["true"] },
        ],
      }),
    );
    state.subnets = Subnets.map((subnet) => subnet.SubnetId);
  }),
  new ScenarioOutput(
    "gotSubnets",
    /**
     * @param {{ subnets: string[] }} state
     */
    (state) =>
      MESSAGES.gotSubnets.replace("${SUBNETS}", state.subnets.join(", ")),
  ),
  new ScenarioOutput(
    "creatingLoadBalancerTargetGroup",
    MESSAGES.creatingLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioAction("createLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { TargetGroups } = await client.send(
      new CreateTargetGroupCommand({
        Name: NAMES.loadBalancerTargetGroupName,
        Protocol: "HTTP",
```

```
        Port: 80,
        HealthCheckPath: "/healthcheck",
        HealthCheckIntervalSeconds: 10,
        HealthCheckTimeoutSeconds: 5,
        HealthyThresholdCount: 2,
        UnhealthyThresholdCount: 2,
        VpcId: state.defaultVpc,
      }),
    );
    const targetGroup = TargetGroups[0];
    state.targetGroupArn = targetGroup.TargetGroupArn;
    state.targetGroupProtocol = targetGroup.Protocol;
    state.targetGroupPort = targetGroup.Port;
  }),
  new ScenarioOutput(
    "createdLoadBalancerTargetGroup",
    MESSAGES.createdLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    ),
  ),
  new ScenarioOutput(
    "creatingLoadBalancer",
    MESSAGES.creatingLoadBalancer.replace("${LB_NAME}", NAMES.loadBalancerName),
  ),
  new ScenarioAction("createLoadBalancer", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { LoadBalancers } = await client.send(
      new CreateLoadBalancerCommand({
        Name: NAMES.loadBalancerName,
        Subnets: state.subnets,
      }),
    );
    state.loadBalancerDns = LoadBalancers[0].DNSName;
    state.loadBalancerArn = LoadBalancers[0].LoadBalancerArn;
    await waitUntilLoadBalancerAvailable(
      { client },
      { Names: [NAMES.loadBalancerName] },
    );
  }),
  new ScenarioOutput("createdLoadBalancer", (state) =>
    MESSAGES.createdLoadBalancer
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${DNS_NAME}", state.loadBalancerDns),
```

```
  ),
  new ScenarioOutput(
    "creatingListener",
    MESSAGES.creatingLoadBalancerListener
      .replace("${LB_NAME}", NAMES.loadBalancerName)
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName),
  ),
  new ScenarioAction("createListener", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    const { Listeners } = await client.send(
      new CreateListenerCommand({
        LoadBalancerArn: state.loadBalancerArn,
        Protocol: state.targetGroupProtocol,
        Port: state.targetGroupPort,
        DefaultActions: [
          { Type: "forward", TargetGroupArn: state.targetGroupArn },
        ],
      }),
    );
    const listener = Listeners[0];
    state.loadBalancerListenerArn = listener.ListenerArn;
  }),
  new ScenarioOutput("createdListener", (state) =>
    MESSAGES.createdLoadBalancerListener.replace(
      "${LB_LISTENER_ARN}",
      state.loadBalancerListenerArn,
    ),
  ),
  new ScenarioOutput(
    "attachingLoadBalancerTargetGroup",
    MESSAGES.attachingLoadBalancerTargetGroup
      .replace("${TARGET_GROUP_NAME}", NAMES.loadBalancerTargetGroupName)
      .replace("${AUTO_SCALING_GROUP_NAME}", NAMES.autoScalingGroupName),
  ),
  new ScenarioAction("attachLoadBalancerTargetGroup", async (state) => {
    const client = new AutoScalingClient({});
    await client.send(
      new AttachLoadBalancerTargetGroupsCommand({
        AutoScalingGroupName: NAMES.autoScalingGroupName,
        TargetGroupARNs: [state.targetGroupArn],
      }),
    );
  }),
  new ScenarioOutput(
```

```
    "attachedLoadBalancerTargetGroup",
    MESSAGES.attachedLoadBalancerTargetGroup,
  ),
  new ScenarioOutput("verifyingInboundPort", MESSAGES.verifyingInboundPort),
  new ScenarioAction(
    "verifyInboundPort",
    /**
     *
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup}} state
     */
    async (state) => {
      const client = new EC2Client({});
      const { SecurityGroups } = await client.send(
        new DescribeSecurityGroupsCommand({
          Filters: [{ Name: "group-name", Values: ["default"] }],
        }),
      );
      if (!SecurityGroups) {
        state.verifyInboundPortError = new Error(MESSAGES.noSecurityGroups);
      }
      state.defaultSecurityGroup = SecurityGroups[0];

      /**
       * @type {string}
       */
      const ipResponse = (await axios.get("http://checkip.amazonaws.com")).data;
      state.myIp = ipResponse.trim();
      const myIpRules = state.defaultSecurityGroup.IpPermissions.filter(
        ({ IpRanges }) =>
          IpRanges.some(
            ({ CidrIp }) =>
              CidrIp.startsWith(state.myIp) || CidrIp === "0.0.0.0/0",
          ),
      )
        .filter(({ IpProtocol }) => IpProtocol === "tcp")
        .filter(({ FromPort }) => FromPort === 80);

      state.myIpRules = myIpRules;
    },
  ),
  new ScenarioOutput(
    "verifiedInboundPort",
    /**
```

```
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return MESSAGES.foundIpRules.replace(
          "${IP_RULES}",
          JSON.stringify(state.myIpRules, null, 2),
        );
      }
      return MESSAGES.noIpRules;
    },
  ),
  new ScenarioInput(
    "shouldAddInboundRule",
    /**
     * @param {{ myIpRules: any[] }} state
     */
    (state) => {
      if (state.myIpRules.length > 0) {
        return false;
      }
      return MESSAGES.noIpRules;
    },
    { type: "confirm" },
  ),
  new ScenarioAction(
    "addInboundRule",
    /**
     * @param {{ defaultSecurityGroup: import('@aws-sdk/client-
ec2').SecurityGroup }} state
     */
    async (state) => {
      if (!state.shouldAddInboundRule) {
        return;
      }

      const client = new EC2Client({});
      await client.send(
        new AuthorizeSecurityGroupIngressCommand({
          GroupId: state.defaultSecurityGroup.GroupId,
          CidrIp: `${state.myIp}/32`,
          FromPort: 80,
          ToPort: 80,
          IpProtocol: "tcp",
```

```
        }),
      );
    },
  ),
  new ScenarioOutput("addedInboundRule", (state) => {
    if (state.shouldAddInboundRule) {
      return MESSAGES.addedInboundRule.replace("${IP_ADDRESS}", state.myIp);
    }
    return false;
  }),
  new ScenarioOutput("verifyingEndpoint", (state) =>
    MESSAGES.verifyingEndpoint.replace("${DNS_NAME}", state.loadBalancerDns),
  ),
  new ScenarioAction("verifyEndpoint", async (state) => {
    try {
      const response = await retry({ intervalInMs: 2000, maxRetries: 30 }, () =>
        axios.get(`http://${state.loadBalancerDns}`),
      );
      state.endpointResponse = JSON.stringify(response.data, null, 2);
    } catch (e) {
      state.verifyEndpointError = e;
    }
  }),
  new ScenarioOutput("verifiedEndpoint", (state) => {
    if (state.verifyEndpointError) {
      console.error(state.verifyEndpointError);
    } else {
      return MESSAGES.verifiedEndpoint.replace(
        "${ENDPOINT_RESPONSE}",
        state.endpointResponse,
      );
    }
  }),
  saveState,
];
```

创建运行演示的步骤。

```
import { readFileSync } from "node:fs";
import { join } from "node:path";

import axios from "axios";
```

```
import {
  DescribeTargetGroupsCommand,
  DescribeTargetHealthCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";
import {
  DescribeInstanceInformationCommand,
  PutParameterCommand,
  SSMClient,
  SendCommandCommand,
} from "@aws-sdk/client-ssm";
import {
  IAMClient,
  CreatePolicyCommand,
  CreateRoleCommand,
  AttachRolePolicyCommand,
  CreateInstanceProfileCommand,
  AddRoleToInstanceProfileCommand,
  waitUntilInstanceProfileExists,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DescribeAutoScalingGroupsCommand,
  TerminateInstanceInAutoScalingGroupCommand,
} from "@aws-sdk/client-auto-scaling";
import {
  DescribeIamInstanceProfileAssociationsCommand,
  EC2Client,
  RebootInstancesCommand,
  ReplaceIamInstanceProfileAssociationCommand,
} from "@aws-sdk/client-ec2";

import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES, RESOURCES_PATH } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

const getRecommendation = new ScenarioAction(
```

```
  "getRecommendation",
  async (state) => {
    const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
    if (loadBalancer) {
      state.loadBalancerDnsName = loadBalancer.DNSName;
      try {
        state.recommendation = (
          await axios.get(`http://${state.loadBalancerDnsName}`)
        ).data;
      } catch (e) {
        state.recommendation = e instanceof Error ? e.message : e;
      }
    } else {
      throw new Error(MESSAGES.demoFindLoadBalancerError);
    }
  },
);

const getRecommendationResult = new ScenarioOutput(
  "getRecommendationResult",
  (state) =>
    `Recommendation:\n${JSON.stringify(state.recommendation, null, 2)}`,
  { preformatted: true },
);

const getHealthCheck = new ScenarioAction("getHealthCheck", async (state) => {
  const client = new ElasticLoadBalancingV2Client({});
  const { TargetGroups } = await client.send(
    new DescribeTargetGroupsCommand({
      Names: [NAMES.loadBalancerTargetGroupName],
    }),
  );

  const { TargetHealthDescriptions } = await client.send(
    new DescribeTargetHealthCommand({
      TargetGroupArn: TargetGroups[0].TargetGroupArn,
    }),
  );
  state.targetHealthDescriptions = TargetHealthDescriptions;
});

const getHealthCheckResult = new ScenarioOutput(
  "getHealthCheckResult",
  /**
```

```
   * @param {{ targetHealthDescriptions: import('@aws-sdk/client-elastic-load-
balancing-v2').TargetHealthDescription[]}} state
   */
  (state) => {
    const status = state.targetHealthDescriptions
      .map((th) => `${th.Target.Id}: ${th.TargetHealth.State}`)
      .join("\n");
    return `Health check:\n${status}`;
  },
  { preformatted: true },
);

const loadBalancerLoop = new ScenarioAction(
  "loadBalancerLoop",
  getRecommendation.action,
  {
    whileConfig: {
      whileFn: ({ loadBalancerCheck }) => loadBalancerCheck,
      input: new ScenarioInput(
        "loadBalancerCheck",
        MESSAGES.demoLoadBalancerCheck,
        {
          type: "confirm",
        },
      ),
      output: getRecommendationResult,
    },
  },
);

const healthCheckLoop = new ScenarioAction(
  "healthCheckLoop",
  getHealthCheck.action,
  {
    whileConfig: {
      whileFn: ({ healthCheck }) => healthCheck,
      input: new ScenarioInput("healthCheck", MESSAGES.demoHealthCheck, {
        type: "confirm",
      }),
      output: getHealthCheckResult,
    },
  },
);
```

```
const statusSteps = [
  getRecommendation,
  getRecommendationResult,
  getHealthCheck,
  getHealthCheckResult,
];

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const demoSteps = [
  new ScenarioOutput("header", MESSAGES.demoHeader, { header: true }),
  new ScenarioOutput("sanityCheck", MESSAGES.demoSanityCheck),
  ...statusSteps,
  new ScenarioInput(
    "brokenDependencyConfirmation",
    MESSAGES.demoBrokenDependencyConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("brokenDependency", async (state) => {
    if (!state.brokenDependencyConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      state.badTableName = `fake-table-${Date.now()}`;
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmTableNameKey,
          Value: state.badTableName,
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testBrokenDependency", (state) =>
    MESSAGES.demoTestBrokenDependency.replace(
      "${TABLE_NAME}",
      state.badTableName,
    ),
  ),
  ...statusSteps,
  new ScenarioInput(
    "staticResponseConfirmation",
```

```
    MESSAGES.demoStaticResponseConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("staticResponse", async (state) => {
    if (!state.staticResponseConfirmation) {
      process.exit();
    } else {
      const client = new SSMClient({});
      await client.send(
        new PutParameterCommand({
          Name: NAMES.ssmFailureResponseKey,
          Value: "static",
          Overwrite: true,
          Type: "String",
        }),
      );
    }
  }),
  new ScenarioOutput("testStaticResponse", MESSAGES.demoTestStaticResponse),
  ...statusSteps,
  new ScenarioInput(
    "badCredentialsConfirmation",
    MESSAGES.demoBadCredentialsConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("badCredentialsExit", (state) => {
    if (!state.badCredentialsConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("fixDynamoDBName", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioAction(
    "badCredentials",
    /**
```

```
     * @param {{ targetInstance: import('@aws-sdk/client-auto-
scaling').Instance }} state
     */
    async (state) => {
      await createSsmOnlyInstanceProfile();
      const autoScalingClient = new AutoScalingClient({});
      const { AutoScalingGroups } = await autoScalingClient.send(
        new DescribeAutoScalingGroupsCommand({
          AutoScalingGroupNames: [NAMES.autoScalingGroupName],
        }),
      );
      state.targetInstance = AutoScalingGroups[0].Instances[0];
      const ec2Client = new EC2Client({});
      const { IamInstanceProfileAssociations } = await ec2Client.send(
        new DescribeIamInstanceProfileAssociationsCommand({
          Filters: [
            { Name: "instance-id", Values: [state.targetInstance.InstanceId] },
          ],
        }),
      );
      state.instanceProfileAssociationId =
        IamInstanceProfileAssociations[0].AssociationId;
      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        ec2Client.send(
          new ReplaceIamInstanceProfileAssociationCommand({
            AssociationId: state.instanceProfileAssociationId,
            IamInstanceProfile: { Name: NAMES.ssmOnlyInstanceProfileName },
          }),
        ),
      );

      await ec2Client.send(
        new RebootInstancesCommand({
          InstanceIds: [state.targetInstance.InstanceId],
        }),
      );

      const ssmClient = new SSMClient({});
      await retry({ intervalInMs: 20000, maxRetries: 15 }, async () => {
        const { InstanceInformationList } = await ssmClient.send(
          new DescribeInstanceInformationCommand({}),
        );

        const instance = InstanceInformationList.find(
```

```
          (info) => info.InstanceId === state.targetInstance.InstanceId,
        );

        if (!instance) {
          throw new Error("Instance not found.");
        }
      });

      await ssmClient.send(
        new SendCommandCommand({
          InstanceIds: [state.targetInstance.InstanceId],
          DocumentName: "AWS-RunShellScript",
          Parameters: { commands: ["cd / && sudo python3 server.py 80"] },
        }),
      );
    },
  ),
  new ScenarioOutput(
    "testBadCredentials",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation}} state
     */
    (state) =>
      MESSAGES.demoTestBadCredentials.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
  ),
  loadBalancerLoop,
  new ScenarioInput(
    "deepHealthCheckConfirmation",
    MESSAGES.demoDeepHealthCheckConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("deepHealthCheckExit", (state) => {
    if (!state.deepHealthCheckConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("deepHealthCheck", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
```

```
        Name: NAMES.ssmHealthCheckKey,
        Value: "deep",
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testDeepHealthCheck", MESSAGES.demoTestDeepHealthCheck),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "killInstanceConfirmation",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    (state) =>
      MESSAGES.demoKillInstanceConfirmation.replace(
        "${INSTANCE_ID}",
        state.targetInstance.InstanceId,
      ),
    { type: "confirm" },
  ),
  new ScenarioAction("killInstanceExit", (state) => {
    if (!state.killInstanceConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction(
    "killInstance",
    /**
     * @param {{ targetInstance: import('@aws-sdk/client-
ssm').InstanceInformation }} state
     */
    async (state) => {
      const client = new AutoScalingClient({});
      await client.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: state.targetInstance.InstanceId,
          ShouldDecrementDesiredCapacity: false,
        }),
      );
    },
  ),
```

```
  new ScenarioOutput("testKillInstance", MESSAGES.demoTestKillInstance),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput("failOpenConfirmation", MESSAGES.demoFailOpenConfirmation, {
    type: "confirm",
  }),
  new ScenarioAction("failOpenExit", (state) => {
    if (!state.failOpenConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("failOpen", () => {
    const client = new SSMClient({});
    return client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: `fake-table-${Date.now()}`,
        Overwrite: true,
        Type: "String",
      }),
    );
  }),
  new ScenarioOutput("testFailOpen", MESSAGES.demoFailOpenTest),
  healthCheckLoop,
  loadBalancerLoop,
  new ScenarioInput(
    "resetTableConfirmation",
    MESSAGES.demoResetTableConfirmation,
    { type: "confirm" },
  ),
  new ScenarioAction("resetTableExit", (state) => {
    if (!state.resetTableConfirmation) {
      process.exit();
    }
  }),
  new ScenarioAction("resetTable", async () => {
    const client = new SSMClient({});
    await client.send(
      new PutParameterCommand({
        Name: NAMES.ssmTableNameKey,
        Value: NAMES.tableName,
        Overwrite: true,
        Type: "String",
      }),
```

```
    );
  }),
  new ScenarioOutput("testResetTable", MESSAGES.demoTestResetTable),
  healthCheckLoop,
  loadBalancerLoop,
];

async function createSsmOnlyInstanceProfile() {
  const iamClient = new IAMClient({});
  const { Policy } = await iamClient.send(
    new CreatePolicyCommand({
      PolicyName: NAMES.ssmOnlyPolicyName,
      PolicyDocument: readFileSync(
        join(RESOURCES_PATH, "ssm_only_policy.json"),
      ),
    }),
  );
  await iamClient.send(
    new CreateRoleCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      AssumeRolePolicyDocument: JSON.stringify({
        Version: "2012-10-17",
        Statement: [
          {
            Effect: "Allow",
            Principal: { Service: "ec2.amazonaws.com" },
            Action: "sts:AssumeRole",
          },
        ],
      }),
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: Policy.Arn,
    }),
  );
  await iamClient.send(
    new AttachRolePolicyCommand({
      RoleName: NAMES.ssmOnlyRoleName,
      PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
    }),
  );
```

```
  const { InstanceProfile } = await iamClient.send(
    new CreateInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
    }),
  );
  await waitUntilInstanceProfileExists(
    { client: iamClient },
    { InstanceProfileName: NAMES.ssmOnlyInstanceProfileName },
  );
  await iamClient.send(
    new AddRoleToInstanceProfileCommand({
      InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
      RoleName: NAMES.ssmOnlyRoleName,
    }),
  );

  return InstanceProfile;
}
```

创建销毁所有资源的步骤。

```
import { unlinkSync } from "node:fs";

import { DynamoDBClient, DeleteTableCommand } from "@aws-sdk/client-dynamodb";
import {
  EC2Client,
  DeleteKeyPairCommand,
  DeleteLaunchTemplateCommand,
  RevokeSecurityGroupIngressCommand,
} from "@aws-sdk/client-ec2";
import {
  IAMClient,
  DeleteInstanceProfileCommand,
  RemoveRoleFromInstanceProfileCommand,
  DeletePolicyCommand,
  DeleteRoleCommand,
  DetachRolePolicyCommand,
  paginateListPolicies,
} from "@aws-sdk/client-iam";
import {
  AutoScalingClient,
  DeleteAutoScalingGroupCommand,
```

```
  TerminateInstanceInAutoScalingGroupCommand,
  UpdateAutoScalingGroupCommand,
  paginateDescribeAutoScalingGroups,
} from "@aws-sdk/client-auto-scaling";
import {
  DeleteLoadBalancerCommand,
  DeleteTargetGroupCommand,
  DescribeTargetGroupsCommand,
  ElasticLoadBalancingV2Client,
} from "@aws-sdk/client-elastic-load-balancing-v2";

import {
  ScenarioOutput,
  ScenarioInput,
  ScenarioAction,
} from "@aws-doc-sdk-examples/lib/scenario/index.js";
import { loadState } from "@aws-doc-sdk-examples/lib/scenario/steps-common.js";
import { retry } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";

import { MESSAGES, NAMES } from "./constants.js";
import { findLoadBalancer } from "./shared.js";

/**
 * @type {import('@aws-doc-sdk-examples/lib/scenario.js').Step[]}
 */
export const destroySteps = [
  loadState,
  new ScenarioInput("destroy", MESSAGES.destroy, { type: "confirm" }),
  new ScenarioAction(
    "abort",
    (state) => state.destroy === false && process.exit(),
  ),
  new ScenarioAction("deleteTable", async (c) => {
    try {
      const client = new DynamoDBClient({});
      await client.send(new DeleteTableCommand({ TableName: NAMES.tableName }));
    } catch (e) {
      c.deleteTableError = e;
    }
  }),
  new ScenarioOutput("deleteTableResult", (state) => {
    if (state.deleteTableError) {
      console.error(state.deleteTableError);
      return MESSAGES.deleteTableError.replace(
```

```
        "${TABLE_NAME}",
        NAMES.tableName,
      );
    }
    return MESSAGES.deletedTable.replace("${TABLE_NAME}", NAMES.tableName);
  }),
  new ScenarioAction("deleteKeyPair", async (state) => {
    try {
      const client = new EC2Client({});
      await client.send(
        new DeleteKeyPairCommand({ KeyName: NAMES.keyPairName }),
      );
      unlinkSync(`${NAMES.keyPairName}.pem`);
    } catch (e) {
      state.deleteKeyPairError = e;
    }
  }),
  new ScenarioOutput("deleteKeyPairResult", (state) => {
    if (state.deleteKeyPairError) {
      console.error(state.deleteKeyPairError);
      return MESSAGES.deleteKeyPairError.replace(
        "${KEY_PAIR_NAME}",
        NAMES.keyPairName,
      );
    }
    return MESSAGES.deletedKeyPair.replace(
      "${KEY_PAIR_NAME}",
      NAMES.keyPairName,
    );
  }),
  new ScenarioAction("detachPolicyFromRole", async (state) => {
    try {
      const client = new IAMClient({});
      const policy = await findPolicy(NAMES.instancePolicyName);

      if (!policy) {
        state.detachPolicyFromRoleError = new Error(
          `Policy ${NAMES.instancePolicyName} not found.`,
        );
      } else {
        await client.send(
          new DetachRolePolicyCommand({
            RoleName: NAMES.instanceRoleName,
            PolicyArn: policy.Arn,
```

```
        }),
      );
    }
  } catch (e) {
    state.detachPolicyFromRoleError = e;
  }
}),
new ScenarioOutput("detachedPolicyFromRole", (state) => {
  if (state.detachPolicyFromRoleError) {
    console.error(state.detachPolicyFromRoleError);
    return MESSAGES.detachPolicyFromRoleError
      .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.detachedPolicyFromRole
    .replace("${INSTANCE_POLICY_NAME}", NAMES.instancePolicyName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstancePolicy", async (state) => {
  const client = new IAMClient({});
  const policy = await findPolicy(NAMES.instancePolicyName);

  if (!policy) {
    state.deletePolicyError = new Error(
      `Policy ${NAMES.instancePolicyName} not found.`,
    );
  } else {
    return client.send(
      new DeletePolicyCommand({
        PolicyArn: policy.Arn,
      }),
    );
  }
}),
new ScenarioOutput("deletePolicyResult", (state) => {
  if (state.deletePolicyError) {
    console.error(state.deletePolicyError);
    return MESSAGES.deletePolicyError.replace(
      "${INSTANCE_POLICY_NAME}",
      NAMES.instancePolicyName,
    );
  }
  return MESSAGES.deletedPolicy.replace(
    "${INSTANCE_POLICY_NAME}",
```

```
    NAMES.instancePolicyName,
  );
}),
new ScenarioAction("removeRoleFromInstanceProfile", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new RemoveRoleFromInstanceProfileCommand({
        RoleName: NAMES.instanceRoleName,
        InstanceProfileName: NAMES.instanceProfileName,
      }),
    );
  } catch (e) {
    state.removeRoleFromInstanceProfileError = e;
  }
}),
new ScenarioOutput("removeRoleFromInstanceProfileResult", (state) => {
  if (state.removeRoleFromInstanceProfile) {
    console.error(state.removeRoleFromInstanceProfileError);
    return MESSAGES.removeRoleFromInstanceProfileError
      .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
      .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
  }
  return MESSAGES.removedRoleFromInstanceProfile
    .replace("${INSTANCE_PROFILE_NAME}", NAMES.instanceProfileName)
    .replace("${INSTANCE_ROLE_NAME}", NAMES.instanceRoleName);
}),
new ScenarioAction("deleteInstanceRole", async (state) => {
  try {
    const client = new IAMClient({});
    await client.send(
      new DeleteRoleCommand({
        RoleName: NAMES.instanceRoleName,
      }),
    );
  } catch (e) {
    state.deleteInstanceRoleError = e;
  }
}),
new ScenarioOutput("deleteInstanceRoleResult", (state) => {
  if (state.deleteInstanceRoleError) {
    console.error(state.deleteInstanceRoleError);
    return MESSAGES.deleteInstanceRoleError.replace(
      "${INSTANCE_ROLE_NAME}",
```

```
        NAMES.instanceRoleName,
      );
    }
    return MESSAGES.deletedInstanceRole.replace(
      "${INSTANCE_ROLE_NAME}",
      NAMES.instanceRoleName,
    );
  }),
  new ScenarioAction("deleteInstanceProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.instanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteInstanceProfileResult", (state) => {
    if (state.deleteInstanceProfileError) {
      console.error(state.deleteInstanceProfileError);
      return MESSAGES.deleteInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.instanceProfileName,
      );
    }
    return MESSAGES.deletedInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.instanceProfileName,
    );
  }),
  new ScenarioAction("deleteAutoScalingGroup", async (state) => {
    try {
      await terminateGroupInstances(NAMES.autoScalingGroupName);
      await retry({ intervalInMs: 60000, maxRetries: 60 }, async () => {
        await deleteAutoScalingGroup(NAMES.autoScalingGroupName);
      });
    } catch (e) {
      state.deleteAutoScalingGroupError = e;
    }
  }),
  new ScenarioOutput("deleteAutoScalingGroupResult", (state) => {
```

```
    if (state.deleteAutoScalingGroupError) {
      console.error(state.deleteAutoScalingGroupError);
      return MESSAGES.deleteAutoScalingGroupError.replace(
        "${AUTO_SCALING_GROUP_NAME}",
        NAMES.autoScalingGroupName,
      );
    }
    return MESSAGES.deletedAutoScalingGroup.replace(
      "${AUTO_SCALING_GROUP_NAME}",
      NAMES.autoScalingGroupName,
    );
  }),
  new ScenarioAction("deleteLaunchTemplate", async (state) => {
    const client = new EC2Client({});
    try {
      await client.send(
        new DeleteLaunchTemplateCommand({
          LaunchTemplateName: NAMES.launchTemplateName,
        }),
      );
    } catch (e) {
      state.deleteLaunchTemplateError = e;
    }
  }),
  new ScenarioOutput("deleteLaunchTemplateResult", (state) => {
    if (state.deleteLaunchTemplateError) {
      console.error(state.deleteLaunchTemplateError);
      return MESSAGES.deleteLaunchTemplateError.replace(
        "${LAUNCH_TEMPLATE_NAME}",
        NAMES.launchTemplateName,
      );
    }
    return MESSAGES.deletedLaunchTemplate.replace(
      "${LAUNCH_TEMPLATE_NAME}",
      NAMES.launchTemplateName,
    );
  }),
  new ScenarioAction("deleteLoadBalancer", async (state) => {
    try {
      const client = new ElasticLoadBalancingV2Client({});
      const loadBalancer = await findLoadBalancer(NAMES.loadBalancerName);
      await client.send(
        new DeleteLoadBalancerCommand({
          LoadBalancerArn: loadBalancer.LoadBalancerArn,
```

```
        }),
      );
      await retry({ intervalInMs: 1000, maxRetries: 60 }, async () => {
        const lb = await findLoadBalancer(NAMES.loadBalancerName);
        if (lb) {
          throw new Error("Load balancer still exists.");
        }
      });
    } catch (e) {
      state.deleteLoadBalancerError = e;
    }
  }),
  new ScenarioOutput("deleteLoadBalancerResult", (state) => {
    if (state.deleteLoadBalancerError) {
      console.error(state.deleteLoadBalancerError);
      return MESSAGES.deleteLoadBalancerError.replace(
        "${LB_NAME}",
        NAMES.loadBalancerName,
      );
    }
    return MESSAGES.deletedLoadBalancer.replace(
      "${LB_NAME}",
      NAMES.loadBalancerName,
    );
  }),
  new ScenarioAction("deleteLoadBalancerTargetGroup", async (state) => {
    const client = new ElasticLoadBalancingV2Client({});
    try {
      const { TargetGroups } = await client.send(
        new DescribeTargetGroupsCommand({
          Names: [NAMES.loadBalancerTargetGroupName],
        }),
      );

      await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
        client.send(
          new DeleteTargetGroupCommand({
            TargetGroupArn: TargetGroups[0].TargetGroupArn,
          }),
        ),
      );
    } catch (e) {
      state.deleteLoadBalancerTargetGroupError = e;
    }
```

```
  }),
  new ScenarioOutput("deleteLoadBalancerTargetGroupResult", (state) => {
    if (state.deleteLoadBalancerTargetGroupError) {
      console.error(state.deleteLoadBalancerTargetGroupError);
      return MESSAGES.deleteLoadBalancerTargetGroupError.replace(
        "${TARGET_GROUP_NAME}",
        NAMES.loadBalancerTargetGroupName,
      );
    }
    return MESSAGES.deletedLoadBalancerTargetGroup.replace(
      "${TARGET_GROUP_NAME}",
      NAMES.loadBalancerTargetGroupName,
    );
  }),
  new ScenarioAction("detachSsmOnlyRoleFromProfile", async (state) => {
    try {
      const client = new IAMClient({});
      await client.send(
        new RemoveRoleFromInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyRoleFromProfileError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyRoleFromProfileResult", (state) => {
    if (state.detachSsmOnlyRoleFromProfileError) {
      console.error(state.detachSsmOnlyRoleFromProfileError);
      return MESSAGES.detachSsmOnlyRoleFromProfileError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
    }
    return MESSAGES.detachedSsmOnlyRoleFromProfile
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${PROFILE_NAME}", NAMES.ssmOnlyInstanceProfileName);
  }),
  new ScenarioAction("detachSsmOnlyCustomRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DetachRolePolicyCommand({
```

```
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.detachSsmOnlyCustomRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyCustomRolePolicyResult", (state) => {
    if (state.detachSsmOnlyCustomRolePolicyError) {
      console.error(state.detachSsmOnlyCustomRolePolicyError);
      return MESSAGES.detachSsmOnlyCustomRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
    }
    return MESSAGES.detachedSsmOnlyCustomRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", NAMES.ssmOnlyPolicyName);
  }),
  new ScenarioAction("detachSsmOnlyAWSRolePolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DetachRolePolicyCommand({
          RoleName: NAMES.ssmOnlyRoleName,
          PolicyArn: "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore",
        }),
      );
    } catch (e) {
      state.detachSsmOnlyAWSRolePolicyError = e;
    }
  }),
  new ScenarioOutput("detachSsmOnlyAWSRolePolicyResult", (state) => {
    if (state.detachSsmOnlyAWSRolePolicyError) {
      console.error(state.detachSsmOnlyAWSRolePolicyError);
      return MESSAGES.detachSsmOnlyAWSRolePolicyError
        .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
        .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
    }
    return MESSAGES.detachedSsmOnlyAWSRolePolicy
      .replace("${ROLE_NAME}", NAMES.ssmOnlyRoleName)
      .replace("${POLICY_NAME}", "AmazonSSMManagedInstanceCore");
  }),
  new ScenarioAction("deleteSsmOnlyInstanceProfile", async (state) => {
```

```
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteInstanceProfileCommand({
          InstanceProfileName: NAMES.ssmOnlyInstanceProfileName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyInstanceProfileError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyInstanceProfileResult", (state) => {
    if (state.deleteSsmOnlyInstanceProfileError) {
      console.error(state.deleteSsmOnlyInstanceProfileError);
      return MESSAGES.deleteSsmOnlyInstanceProfileError.replace(
        "${INSTANCE_PROFILE_NAME}",
        NAMES.ssmOnlyInstanceProfileName,
      );
    }
    return MESSAGES.deletedSsmOnlyInstanceProfile.replace(
      "${INSTANCE_PROFILE_NAME}",
      NAMES.ssmOnlyInstanceProfileName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyPolicy", async (state) => {
    try {
      const iamClient = new IAMClient({});
      const ssmOnlyPolicy = await findPolicy(NAMES.ssmOnlyPolicyName);
      await iamClient.send(
        new DeletePolicyCommand({
          PolicyArn: ssmOnlyPolicy.Arn,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyPolicyError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyPolicyResult", (state) => {
    if (state.deleteSsmOnlyPolicyError) {
      console.error(state.deleteSsmOnlyPolicyError);
      return MESSAGES.deleteSsmOnlyPolicyError.replace(
        "${POLICY_NAME}",
        NAMES.ssmOnlyPolicyName,
      );
```

```
    }
    return MESSAGES.deletedSsmOnlyPolicy.replace(
      "${POLICY_NAME}",
      NAMES.ssmOnlyPolicyName,
    );
  }),
  new ScenarioAction("deleteSsmOnlyRole", async (state) => {
    try {
      const iamClient = new IAMClient({});
      await iamClient.send(
        new DeleteRoleCommand({
          RoleName: NAMES.ssmOnlyRoleName,
        }),
      );
    } catch (e) {
      state.deleteSsmOnlyRoleError = e;
    }
  }),
  new ScenarioOutput("deleteSsmOnlyRoleResult", (state) => {
    if (state.deleteSsmOnlyRoleError) {
      console.error(state.deleteSsmOnlyRoleError);
      return MESSAGES.deleteSsmOnlyRoleError.replace(
        "${ROLE_NAME}",
        NAMES.ssmOnlyRoleName,
      );
    }
    return MESSAGES.deletedSsmOnlyRole.replace(
      "${ROLE_NAME}",
      NAMES.ssmOnlyRoleName,
    );
  }),
  new ScenarioAction(
    "revokeSecurityGroupIngress",
    async (
      /** @type {{ myIp: string, defaultSecurityGroup: { GroupId: string } }} */
state,
    ) => {
      const ec2Client = new EC2Client({});

      try {
        await ec2Client.send(
          new RevokeSecurityGroupIngressCommand({
            GroupId: state.defaultSecurityGroup.GroupId,
            CidrIp: `${state.myIp}/32`,
```

```
            FromPort: 80,
            ToPort: 80,
            IpProtocol: "tcp",
          }),
        );
      } catch (e) {
        state.revokeSecurityGroupIngressError = e;
      }
    },
  ),
  new ScenarioOutput("revokeSecurityGroupIngressResult", (state) => {
    if (state.revokeSecurityGroupIngressError) {
      console.error(state.revokeSecurityGroupIngressError);
      return MESSAGES.revokeSecurityGroupIngressError.replace(
        "${IP}",
        state.myIp,
      );
    }
    return MESSAGES.revokedSecurityGroupIngress.replace("${IP}", state.myIp);
  }),
];

/**
 * @param {string} policyName
 */
async function findPolicy(policyName) {
  const client = new IAMClient({});
  const paginatedPolicies = paginateListPolicies({ client }, {});
  for await (const page of paginatedPolicies) {
    const policy = page.Policies.find((p) => p.PolicyName === policyName);
    if (policy) {
      return policy;
    }
  }
}

/**
 * @param {string} groupName
 */
async function deleteAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  try {
    await client.send(
      new DeleteAutoScalingGroupCommand({
```

```
        AutoScalingGroupName: groupName,
      }),
    );
  } catch (err) {
    if (!(err instanceof Error)) {
      throw err;
    }
    console.log(err.name);
    throw err;
  }
}

/**
 * @param {string} groupName
 */
async function terminateGroupInstances(groupName) {
  const autoScalingClient = new AutoScalingClient({});
  const group = await findAutoScalingGroup(groupName);
  await autoScalingClient.send(
    new UpdateAutoScalingGroupCommand({
      AutoScalingGroupName: group.AutoScalingGroupName,
      MinSize: 0,
    }),
  );
  for (const i of group.Instances) {
    await retry({ intervalInMs: 1000, maxRetries: 30 }, () =>
      autoScalingClient.send(
        new TerminateInstanceInAutoScalingGroupCommand({
          InstanceId: i.InstanceId,
          ShouldDecrementDesiredCapacity: true,
        }),
      ),
    );
  }
}

async function findAutoScalingGroup(groupName) {
  const client = new AutoScalingClient({});
  const paginatedGroups = paginateDescribeAutoScalingGroups({ client }, {});
  for await (const page of paginatedGroups) {
    const group = page.AutoScalingGroups.find(
      (g) => g.AutoScalingGroupName === groupName,
    );
    if (group) {
```

```
      return group;
    }
  }
  throw new Error(`Auto scaling group ${groupName} not found.`);
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的以下主题。
  - AttachLoadBalancerTargetGroups
  - CreateAutoScalingGroup
  - CreateInstanceProfile
  - CreateLaunchTemplate
  - CreateListener
  - CreateLoadBalancer
  - CreateTargetGroup
  - DeleteAutoScalingGroup
  - DeleteInstanceProfile
  - DeleteLaunchTemplate
  - DeleteLoadBalancer
  - DeleteTargetGroup
  - DescribeAutoScalingGroups
  - DescribeAvailabilityZones
  - DescribeIamInstanceProfileAssociations
  - DescribeInstances
  - DescribeLoadBalancers
  - DescribeSubnets
  - DescribeTargetGroups
  - DescribeTargetHealth
  - DescribeVpcs
  - RebootInstances
  - ReplaceIamInstanceProfileAssociation
  - TerminateInstanceInAutoScalingGroup

- UpdateAutoScalingGroup

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
class Runner:
    """
    Manages the deployment, demonstration, and destruction of resources for the
 resilient service.
    """

    def __init__(
        self,
        resource_path: str,
        recommendation: RecommendationService,
        autoscaler: AutoScalingWrapper,
        loadbalancer: ElasticLoadBalancerWrapper,
        param_helper: ParameterHelper,
    ):
        """
        Initializes the Runner class with the necessary parameters.

        :param resource_path: The path to resource files used by this example,
 such as IAM policies and instance scripts.
        :param recommendation: An instance of the RecommendationService class.
        :param autoscaler: An instance of the AutoScaler class.
        :param loadbalancer: An instance of the LoadBalancer class.
        :param param_helper: An instance of the ParameterHelper class.
        """
        self.resource_path = resource_path
        self.recommendation = recommendation
        self.autoscaler = autoscaler
        self.loadbalancer = loadbalancer
```

```
        self.param_helper = param_helper
        self.protocol = "HTTP"
        self.port = 80
        self.ssh_port = 22

        prefix = "doc-example-resilience"
        self.target_group_name = f"{prefix}-tg"
        self.load_balancer_name = f"{prefix}-lb"

    def deploy(self) -> None:
        """
        Deploys the resources required for the resilient service, including the
DynamoDB table,
        EC2 instances, Auto Scaling group, and load balancer.
        """
        recommendations_path = f"{self.resource_path}/recommendations.json"
        startup_script = f"{self.resource_path}/server_startup_script.sh"
        instance_policy = f"{self.resource_path}/instance_policy.json"

        logging.info("Starting deployment of resources for the resilient
service.")

        logging.info(
            "Creating and populating DynamoDB table '%s'.",
            self.recommendation.table_name,
        )
        self.recommendation.create()
        self.recommendation.populate(recommendations_path)

        logging.info(
            "Creating an EC2 launch template with the startup script '%s'.",
            startup_script,
        )
        self.autoscaler.create_template(startup_script, instance_policy)

        logging.info(
            "Creating an EC2 Auto Scaling group across multiple Availability
Zones."
        )
        zones = self.autoscaler.create_autoscaling_group(3)

        logging.info("Creating variables that control the flow of the demo.")
        self.param_helper.reset()
```

```
        logging.info("Creating Elastic Load Balancing target group and load
 balancer.")

        vpc = self.autoscaler.get_default_vpc()
        subnets = self.autoscaler.get_subnets(vpc["VpcId"], zones)
        target_group = self.loadbalancer.create_target_group(
            self.target_group_name, self.protocol, self.port, vpc["VpcId"]
        )
        self.loadbalancer.create_load_balancer(
            self.load_balancer_name, [subnet["SubnetId"] for subnet in subnets]
        )
        self.loadbalancer.create_listener(self.load_balancer_name, target_group)

        self.autoscaler.attach_load_balancer_target_group(target_group)

        logging.info("Verifying access to the load balancer endpoint.")
        endpoint = self.loadbalancer.get_endpoint(self.load_balancer_name)
        lb_success = self.loadbalancer.verify_load_balancer_endpoint(endpoint)
        current_ip_address = requests.get("http://
checkip.amazonaws.com").text.strip()

        if not lb_success:
            logging.warning(
                "Couldn't connect to the load balancer. Verifying that the port
 is open..."
            )
            sec_group, port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.port, current_ip_address
            )
            sec_group, ssh_port_is_open = self.autoscaler.verify_inbound_port(
                vpc, self.ssh_port, current_ip_address
            )
            if not port_is_open:
                logging.warning(
                    "The default security group for your VPC must allow access
 from this computer."
                )
                if q.ask(
                    f"Do you want to add a rule to security group
 {sec_group['GroupId']} to allow\n"
                    f"inbound traffic on port {self.port} from your computer's IP
 address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
```

```
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.port, current_ip_address
                    )
            if not ssh_port_is_open:
                if q.ask(
                    f"Do you want to add a rule to security group
{sec_group['GroupId']} to allow\n"
                    f"inbound SSH traffic on port {self.ssh_port} for debugging
from your computer's IP address of {current_ip_address}? (y/n) ",
                    q.is_yesno,
                ):
                    self.autoscaler.open_inbound_port(
                        sec_group["GroupId"], self.ssh_port, current_ip_address
                    )
            lb_success =
self.loadbalancer.verify_load_balancer_endpoint(endpoint)

        if lb_success:
            logging.info(
                "Load balancer is ready. Access it at: http://%s",
current_ip_address
            )
        else:
            logging.error(
                "Couldn't get a successful response from the load balancer
endpoint. Please verify your VPC and security group settings."
            )

    def demo_choices(self) -> None:
        """
        Presents choices for interacting with the deployed service, such as
sending requests to
        the load balancer or checking the health of the targets.
        """
        actions = [
            "Send a GET request to the load balancer endpoint.",
            "Check the health of load balancer targets.",
            "Go to the next part of the demo.",
        ]
        choice = 0
        while choice != 2:
            logging.info("Choose an action to interact with the service.")
            choice = q.choose("Which action would you like to take? ", actions)
            if choice == 0:
```

```
                logging.info("Sending a GET request to the load balancer
endpoint.")
                endpoint =
self.loadbalancer.get_endpoint(self.load_balancer_name)
                logging.info("GET http://%s", endpoint)
                response = requests.get(f"http://{endpoint}")
                logging.info("Response: %s", response.status_code)
                if response.headers.get("content-type") == "application/json":
                    pp(response.json())
            elif choice == 1:
                logging.info("Checking the health of load balancer targets.")
                health =
self.loadbalancer.check_target_health(self.target_group_name)
                for target in health:
                    state = target["TargetHealth"]["State"]
                    logging.info(
                        "Target %s on port %d is %s",
                        target["Target"]["Id"],
                        target["Target"]["Port"],
                        state,
                    )
                    if state != "healthy":
                        logging.warning(
                            "%s: %s",
                            target["TargetHealth"]["Reason"],
                            target["TargetHealth"]["Description"],
                        )
                logging.info(
                    "Note that it can take a minute or two for the health check
to update."
                )
            elif choice == 2:
                logging.info("Proceeding to the next part of the demo.")

    def demo(self) -> None:
        """
        Runs the demonstration, showing how the service responds to different
failure scenarios
        and how a resilient architecture can keep the service running.
        """
        ssm_only_policy = f"{self.resource_path}/ssm_only_policy.json"

        logging.info("Resetting parameters to starting values for the demo.")
        self.param_helper.reset()
```

```
        logging.info(
            "Starting demonstration of the service's resilience under various
failure conditions."
        )
        self.demo_choices()

        logging.info(
            "Simulating failure by changing the Systems Manager parameter to a
non-existent table."
        )
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info("Sending GET requests will now return failure codes.")
        self.demo_choices()

        logging.info("Switching to static response mode to mitigate failure.")
        self.param_helper.put(self.param_helper.failure_response, "static")
        logging.info("Sending GET requests will now return static responses.")
        self.demo_choices()

        logging.info("Restoring normal operation of the recommendation service.")
        self.param_helper.put(self.param_helper.table,
self.recommendation.table_name)

        logging.info(
            "Introducing a failure by assigning bad credentials to one of the
instances."
        )
        self.autoscaler.create_instance_profile(
            ssm_only_policy,
            self.autoscaler.bad_creds_policy_name,
            self.autoscaler.bad_creds_role_name,
            self.autoscaler.bad_creds_profile_name,
            ["AmazonSSMManagedInstanceCore"],
        )
        instances = self.autoscaler.get_instances()
        bad_instance_id = instances[0]
        instance_profile = self.autoscaler.get_instance_profile(bad_instance_id)
        logging.info(
            "Replacing instance profile with bad credentials for instance %s.",
            bad_instance_id,
        )
        self.autoscaler.replace_instance_profile(
            bad_instance_id,
```

```
            self.autoscaler.bad_creds_profile_name,
            instance_profile["AssociationId"],
        )
        logging.info(
            "Sending GET requests may return either a valid recommendation or a
static response."
        )
        self.demo_choices()

        logging.info("Implementing deep health checks to detect unhealthy
instances.")
        self.param_helper.put(self.param_helper.health_check, "deep")
        logging.info("Checking the health of the load balancer targets.")
        self.demo_choices()

        logging.info(
            "Terminating the unhealthy instance to let the auto scaler replace
it."
        )
        self.autoscaler.terminate_instance(bad_instance_id)
        logging.info("The service remains resilient during instance
replacement.")
        self.demo_choices()

        logging.info("Simulating a complete failure of the recommendation
service.")
        self.param_helper.put(self.param_helper.table, "this-is-not-a-table")
        logging.info(
            "All instances will report as unhealthy, but the service will still
return static responses."
        )
        self.demo_choices()
        self.param_helper.reset()

    def destroy(self, automation=False) -> None:
        """
        Destroys all resources created for the demo, including the load balancer,
Auto Scaling group,
        EC2 instances, and DynamoDB table.
        """
        logging.info(
            "This concludes the demo. Preparing to clean up all AWS resources
created during the demo."
        )
```

```
        if automation:
            cleanup = True
        else:
            cleanup = q.ask(
                "Do you want to clean up all demo resources? (y/n) ", q.is_yesno
            )

        if cleanup:
            logging.info("Deleting load balancer and related resources.")
            self.loadbalancer.delete_load_balancer(self.load_balancer_name)
            self.loadbalancer.delete_target_group(self.target_group_name)
            self.autoscaler.delete_autoscaling_group(self.autoscaler.group_name)
            self.autoscaler.delete_key_pair()
            self.autoscaler.delete_template()
            self.autoscaler.delete_instance_profile(
                self.autoscaler.bad_creds_profile_name,
                self.autoscaler.bad_creds_role_name,
            )
            logging.info("Deleting DynamoDB table and other resources.")
            self.recommendation.destroy()
        else:
            logging.warning(
                "Resources have not been deleted. Ensure you clean them up
 manually to avoid unexpected charges."
            )


def main() -> None:
    """
    Main function to parse arguments and run the appropriate actions for the
 demo.
    """
    parser = argparse.ArgumentParser()
    parser.add_argument(
        "--action",
        required=True,
        choices=["all", "deploy", "demo", "destroy"],
        help="The action to take for the demo. When 'all' is specified, resources
 are\n"
        "deployed, the demo is run, and resources are destroyed.",
    )
    parser.add_argument(
        "--resource_path",
        default="../../../scenarios/features/resilient_service/resources",
```

```
        help="The path to resource files used by this example, such as IAM
policies and\n"
        "instance scripts.",
    )
    args = parser.parse_args()

    logging.info("Starting the Resilient Service demo.")

    prefix = "doc-example-resilience"

    # Service Clients
    ddb_client = boto3.client("dynamodb")
    elb_client = boto3.client("elbv2")
    autoscaling_client = boto3.client("autoscaling")
    ec2_client = boto3.client("ec2")
    ssm_client = boto3.client("ssm")
    iam_client = boto3.client("iam")

    # Wrapper instantiations
    recommendation = RecommendationService(
        "doc-example-recommendation-service", ddb_client
    )
    autoscaling_wrapper = AutoScalingWrapper(
        prefix,
        "t3.micro",
        "/aws/service/ami-amazon-linux-latest/amzn2-ami-hvm-x86_64-gp2",
        autoscaling_client,
        ec2_client,
        ssm_client,
        iam_client,
    )
    elb_wrapper = ElasticLoadBalancerWrapper(elb_client)
    param_helper = ParameterHelper(recommendation.table_name, ssm_client)

    # Demo invocation
    runner = Runner(
        args.resource_path,
        recommendation,
        autoscaling_wrapper,
        elb_wrapper,
        param_helper,
    )
    actions = [args.action] if args.action != "all" else ["deploy", "demo",
"destroy"]
```

```
    for action in actions:
        if action == "deploy":
            runner.deploy()
        elif action == "demo":
            runner.demo()
        elif action == "destroy":
            runner.destroy()

    logging.info("Demo completed successfully.")


if __name__ == "__main__":
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    main()
```

创建一个包含自动扩缩和 Amazon EC2 操作的类。

```
class AutoScalingWrapper:
    """
    Encapsulates Amazon EC2 Auto Scaling and EC2 management actions.
    """

    def __init__(
        self,
        resource_prefix: str,
        inst_type: str,
        ami_param: str,
        autoscaling_client: boto3.client,
        ec2_client: boto3.client,
        ssm_client: boto3.client,
        iam_client: boto3.client,
    ):
        """
        Initializes the AutoScaler class with the necessary parameters.

        :param resource_prefix: The prefix for naming AWS resources that are
 created by this class.
        :param inst_type: The type of EC2 instance to create, such as t3.micro.
        :param ami_param: The Systems Manager parameter used to look up the AMI
 that is created.
        :param autoscaling_client: A Boto3 EC2 Auto Scaling client.
        :param ec2_client: A Boto3 EC2 client.
```

```
        :param ssm_client: A Boto3 Systems Manager client.
        :param iam_client: A Boto3 IAM client.
        """
        self.inst_type = inst_type
        self.ami_param = ami_param
        self.autoscaling_client = autoscaling_client
        self.ec2_client = ec2_client
        self.ssm_client = ssm_client
        self.iam_client = iam_client
        sts_client = boto3.client("sts")
        self.account_id = sts_client.get_caller_identity()["Account"]

        self.key_pair_name = f"{resource_prefix}-key-pair"
        self.launch_template_name = f"{resource_prefix}-template-"
        self.group_name = f"{resource_prefix}-group"

        # Happy path
        self.instance_policy_name = f"{resource_prefix}-pol"
        self.instance_role_name = f"{resource_prefix}-role"
        self.instance_profile_name = f"{resource_prefix}-prof"

        # Failure mode
        self.bad_creds_policy_name = f"{resource_prefix}-bc-pol"
        self.bad_creds_role_name = f"{resource_prefix}-bc-role"
        self.bad_creds_profile_name = f"{resource_prefix}-bc-prof"


    def create_policy(self, policy_file: str, policy_name: str) -> str:
        """
        Creates a new IAM policy or retrieves the ARN of an existing policy.

        :param policy_file: The path to a JSON file that contains the policy
definition.
        :param policy_name: The name to give the created policy.
        :return: The ARN of the created or existing policy.
        """
        with open(policy_file) as file:
            policy_doc = file.read()

        try:
            response = self.iam_client.create_policy(
                PolicyName=policy_name, PolicyDocument=policy_doc
            )
            policy_arn = response["Policy"]["Arn"]
```

```
            log.info(f"Policy '{policy_name}' created successfully. ARN:
 {policy_arn}")
            return policy_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the policy already exists, get its ARN
                response = self.iam_client.get_policy(
                    PolicyArn=f"arn:aws:iam::{self.account_id}:policy/
{policy_name}"
                )
                policy_arn = response["Policy"]["Arn"]
                log.info(f"Policy '{policy_name}' already exists. ARN:
 {policy_arn}")
                return policy_arn
            log.error(f"Full error:\n\t{err}")

    def create_role(self, role_name: str, assume_role_doc: dict) -> str:
        """
        Creates a new IAM role or retrieves the ARN of an existing role.

        :param role_name: The name to give the created role.
        :param assume_role_doc: The assume role policy document that specifies
 which
                                entities can assume the role.
        :return: The ARN of the created or existing role.
        """
        try:
            response = self.iam_client.create_role(
                RoleName=role_name,
 AssumeRolePolicyDocument=json.dumps(assume_role_doc)
            )
            role_arn = response["Role"]["Arn"]
            log.info(f"Role '{role_name}' created successfully. ARN: {role_arn}")
            return role_arn

        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                # If the role already exists, get its ARN
                response = self.iam_client.get_role(RoleName=role_name)
                role_arn = response["Role"]["Arn"]
                log.info(f"Role '{role_name}' already exists. ARN: {role_arn}")
                return role_arn
            log.error(f"Full error:\n\t{err}")
```

```
    def attach_policy(
        self,
        role_name: str,
        policy_arn: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> None:
        """
        Attaches an IAM policy to a role and optionally attaches additional AWS-
managed policies.

        :param role_name: The name of the role to attach the policy to.
        :param policy_arn: The ARN of the policy to attach.
        :param aws_managed_policies: A tuple of AWS-managed policy names to
 attach to the role.
        """
        try:
            self.iam_client.attach_role_policy(RoleName=role_name,
 PolicyArn=policy_arn)
            for aws_policy in aws_managed_policies:
                self.iam_client.attach_role_policy(
                    RoleName=role_name,
                    PolicyArn=f"arn:aws:iam::aws:policy/{aws_policy}",
                )
            log.info(f"Attached policy {policy_arn} to role {role_name}.")
        except ClientError as err:
            log.error(f"Failed to attach policy {policy_arn} to role
 {role_name}.")
            log.error(f"Full error:\n\t{err}")

    def create_instance_profile(
        self,
        policy_file: str,
        policy_name: str,
        role_name: str,
        profile_name: str,
        aws_managed_policies: Tuple[str, ...] = (),
    ) -> str:
        """
        Creates a policy, role, and profile that is associated with instances
 created by
        this class. An instance's associated profile defines a role that is
 assumed by the
```

```
        instance. The role has attached policies that specify the AWS permissions
granted to
        clients that run on the instance.

        :param policy_file: The name of a JSON file that contains the policy
definition to
                            create and attach to the role.
        :param policy_name: The name to give the created policy.
        :param role_name: The name to give the created role.
        :param profile_name: The name to the created profile.
        :param aws_managed_policies: Additional AWS-managed policies that are
attached to
                                     the role, such as
AmazonSSMManagedInstanceCore to grant
                                     use of Systems Manager to send commands to
the instance.
        :return: The ARN of the profile that is created.
        """
        assume_role_doc = {
            "Version":"2012-10-17",
            "Statement": [
                {
                    "Effect": "Allow",
                    "Principal": {"Service": "ec2.amazonaws.com"},
                    "Action": "sts:AssumeRole",
                }
            ],
        }
        policy_arn = self.create_policy(policy_file, policy_name)
        self.create_role(role_name, assume_role_doc)
        self.attach_policy(role_name, policy_arn, aws_managed_policies)

        try:
            profile_response = self.iam_client.create_instance_profile(
                InstanceProfileName=profile_name
            )
            waiter = self.iam_client.get_waiter("instance_profile_exists")
            waiter.wait(InstanceProfileName=profile_name)
            time.sleep(10)  # wait a little longer
            profile_arn = profile_response["InstanceProfile"]["Arn"]
            self.iam_client.add_role_to_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )
```

```
            log.info("Created profile %s and added role %s.", profile_name,
role_name)
        except ClientError as err:
            if err.response["Error"]["Code"] == "EntityAlreadyExists":
                prof_response = self.iam_client.get_instance_profile(
                    InstanceProfileName=profile_name
                )
                profile_arn = prof_response["InstanceProfile"]["Arn"]
                log.info(
                    "Instance profile %s already exists, nothing to do.",
profile_name
                )
            log.error(f"Full error:\n\t{err}")
        return profile_arn


    def get_instance_profile(self, instance_id: str) -> Dict[str, Any]:
        """
        Gets data about the profile associated with an instance.

        :param instance_id: The ID of the instance to look up.
        :return: The profile data.
        """
        try:
            response =
self.ec2_client.describe_iam_instance_profile_associations(
                Filters=[{"Name": "instance-id", "Values": [instance_id]}]
            )
            if not response["IamInstanceProfileAssociations"]:
                log.info(f"No instance profile found for instance
{instance_id}.")
            profile_data = response["IamInstanceProfileAssociations"][0]
            log.info(f"Retrieved instance profile for instance {instance_id}.")
            return profile_data
        except ClientError as err:
            log.error(
                f"Failed to retrieve instance profile for instance
{instance_id}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidInstanceID.NotFound":
                log.error(f"The instance ID '{instance_id}' does not exist.")
            log.error(f"Full error:\n\t{err}")
```

```
   def replace_instance_profile(
       self,
       instance_id: str,
       new_instance_profile_name: str,
       profile_association_id: str,
   ) -> None:
       """
       Replaces the profile associated with a running instance. After the
profile is
       replaced, the instance is rebooted to ensure that it uses the new
profile. When
       the instance is ready, Systems Manager is used to restart the Python web
server.

       :param instance_id: The ID of the instance to restart.
       :param new_instance_profile_name: The name of the new profile to
associate with
                                         the specified instance.
       :param profile_association_id: The ID of the existing profile association
for the
                                      instance.
       """
       try:
           self.ec2_client.replace_iam_instance_profile_association(
               IamInstanceProfile={"Name": new_instance_profile_name},
               AssociationId=profile_association_id,
           )
           log.info(
               "Replaced instance profile for association %s with profile %s.",
               profile_association_id,
               new_instance_profile_name,
           )
           time.sleep(5)

           self.ec2_client.reboot_instances(InstanceIds=[instance_id])
           log.info("Rebooting instance %s.", instance_id)
           waiter = self.ec2_client.get_waiter("instance_running")
           log.info("Waiting for instance %s to be running.", instance_id)
           waiter.wait(InstanceIds=[instance_id])
           log.info("Instance %s is now running.", instance_id)

           self.ssm_client.send_command(
               InstanceIds=[instance_id],
```

```
                DocumentName="AWS-RunShellScript",
                Parameters={"commands": ["cd / && sudo python3 server.py 80"]},
            )
            log.info(f"Restarted the Python web server on instance
'{instance_id}'.")
        except ClientError as err:
            log.error("Failed to replace instance profile.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidAssociationID.NotFound":
                log.error(
                    f"Association ID '{profile_association_id}' does not exist."
                    "Please check the association ID and try again."
                )
            if error_code == "InvalidInstanceId":
                log.error(
                    f"The specified instance ID '{instance_id}' does not exist or
is not available for SSM. "
                    f"Please verify the instance ID and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_instance_profile(self, profile_name: str, role_name: str) -> None:
        """
        Detaches a role from an instance profile, detaches policies from the
role,
        and deletes all the resources.

        :param profile_name: The name of the profile to delete.
        :param role_name: The name of the role to delete.
        """
        try:
            self.iam_client.remove_role_from_instance_profile(
                InstanceProfileName=profile_name, RoleName=role_name
            )

self.iam_client.delete_instance_profile(InstanceProfileName=profile_name)
            log.info("Deleted instance profile %s.", profile_name)
            attached_policies = self.iam_client.list_attached_role_policies(
                RoleName=role_name
            )
            for pol in attached_policies["AttachedPolicies"]:
                self.iam_client.detach_role_policy(
                    RoleName=role_name, PolicyArn=pol["PolicyArn"]
```

```
                )
                if not pol["PolicyArn"].startswith("arn:aws:iam::aws"):
                    self.iam_client.delete_policy(PolicyArn=pol["PolicyArn"])
                log.info("Detached and deleted policy %s.", pol["PolicyName"])
            self.iam_client.delete_role(RoleName=role_name)
            log.info("Deleted role %s.", role_name)
        except ClientError as err:
            log.error(
                f"Couldn't delete instance profile {profile_name} or detach "
                f"policies and delete role {role_name}: {err}"
            )
            if err.response["Error"]["Code"] == "NoSuchEntity":
                log.info(
                    "Instance profile %s doesn't exist, nothing to do.",
profile_name
                )


    def create_key_pair(self, key_pair_name: str) -> None:
        """
        Creates a new key pair.

        :param key_pair_name: The name of the key pair to create.
        """
        try:
            response = self.ec2_client.create_key_pair(KeyName=key_pair_name)
            with open(f"{key_pair_name}.pem", "w") as file:
                file.write(response["KeyMaterial"])
            chmod(f"{key_pair_name}.pem", 0o600)
            log.info("Created key pair %s.", key_pair_name)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to create key pair {key_pair_name}.")
            if error_code == "InvalidKeyPair.Duplicate":
                log.error(f"A key pair with the name '{key_pair_name}' already
exists.")
            log.error(f"Full error:\n\t{err}")


    def delete_key_pair(self) -> None:
        """
        Deletes a key pair.
        """
        try:
```

```
            self.ec2_client.delete_key_pair(KeyName=self.key_pair_name)
            remove(f"{self.key_pair_name}.pem")
            log.info("Deleted key pair %s.", self.key_pair_name)
        except ClientError as err:
            log.error(f"Couldn't delete key pair '{self.key_pair_name}'.")
            log.error(f"Full error:\n\t{err}")
        except FileNotFoundError as err:
            log.info("Key pair %s doesn't exist, nothing to do.",
self.key_pair_name)
            log.error(f"Full error:\n\t{err}")


    def create_template(
        self, server_startup_script_file: str, instance_policy_file: str
    ) -> Dict[str, Any]:
        """
        Creates an Amazon EC2 launch template to use with Amazon EC2 Auto
Scaling. The
        launch template specifies a Bash script in its user data field that runs
after
        the instance is started. This script installs Python packages and starts
a
        Python web server on the instance.

        :param server_startup_script_file: The path to a Bash script file that is
run
                                           when an instance starts.
        :param instance_policy_file: The path to a file that defines a
permissions policy
                                     to create and attach to the instance
profile.
        :return: Information about the newly created template.
        """
        template = {}
        try:
            # Create key pair and instance profile
            self.create_key_pair(self.key_pair_name)
            self.create_instance_profile(
                instance_policy_file,
                self.instance_policy_name,
                self.instance_role_name,
                self.instance_profile_name,
            )
```

```
            # Read the startup script
            with open(server_startup_script_file) as file:
                start_server_script = file.read()

            # Get the latest AMI ID
            ami_latest = self.ssm_client.get_parameter(Name=self.ami_param)
            ami_id = ami_latest["Parameter"]["Value"]

            # Create the launch template
            lt_response = self.ec2_client.create_launch_template(
                LaunchTemplateName=self.launch_template_name,
                LaunchTemplateData={
                    "InstanceType": self.inst_type,
                    "ImageId": ami_id,
                    "IamInstanceProfile": {"Name": self.instance_profile_name},
                    "UserData": base64.b64encode(
                        start_server_script.encode(encoding="utf-8")
                    ).decode(encoding="utf-8"),
                    "KeyName": self.key_pair_name,
                },
            )
            template = lt_response["LaunchTemplate"]
            log.info(
                f"Created launch template {self.launch_template_name} for AMI
{ami_id} on {self.inst_type}."
            )
        except ClientError as err:
            log.error(f"Failed to create launch template
{self.launch_template_name}.")
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidLaunchTemplateName.AlreadyExistsException":
                log.info(
                    f"Launch template {self.launch_template_name} already exists,
nothing to do."
                )
            log.error(f"Full error:\n\t{err}")
        return template


    def delete_template(self):
        """
        Deletes a launch template.
        """
        try:
```

```
            self.ec2_client.delete_launch_template(
                LaunchTemplateName=self.launch_template_name
            )
            self.delete_instance_profile(
                self.instance_profile_name, self.instance_role_name
            )
            log.info("Launch template %s deleted.", self.launch_template_name)
        except ClientError as err:
            if (
                err.response["Error"]["Code"]
                == "InvalidLaunchTemplateName.NotFoundException"
            ):
                log.info(
                    "Launch template %s does not exist, nothing to do.",
                    self.launch_template_name,
                )
            log.error(f"Full error:\n\t{err}")


    def get_availability_zones(self) -> List[str]:
        """
        Gets a list of Availability Zones in the AWS Region of the Amazon EC2
client.

        :return: The list of Availability Zones for the client Region.
        """
        try:
            response = self.ec2_client.describe_availability_zones()
            zones = [zone["ZoneName"] for zone in response["AvailabilityZones"]]
            log.info(f"Retrieved {len(zones)} availability zones: {zones}.")
        except ClientError as err:
            log.error("Failed to retrieve availability zones.")
            log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def create_autoscaling_group(self, group_size: int) -> List[str]:
        """
        Creates an EC2 Auto Scaling group with the specified size.

        :param group_size: The number of instances to set for the minimum and
maximum in
                           the group.
```

```
        :return: The list of Availability Zones specified for the group.
        """
        try:
            zones = self.get_availability_zones()
            self.autoscaling_client.create_auto_scaling_group(
                AutoScalingGroupName=self.group_name,
                AvailabilityZones=zones,
                LaunchTemplate={
                    "LaunchTemplateName": self.launch_template_name,
                    "Version": "$Default",
                },
                MinSize=group_size,
                MaxSize=group_size,
            )
            log.info(
                f"Created EC2 Auto Scaling group {self.group_name} with
availability zones {zones}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            if error_code == "AlreadyExists":
                log.info(
                    f"EC2 Auto Scaling group {self.group_name} already exists,
nothing to do."
                )
            else:
                log.error(f"Failed to create EC2 Auto Scaling group
{self.group_name}.")
                log.error(f"Full error:\n\t{err}")
        else:
            return zones


    def get_instances(self) -> List[str]:
        """
        Gets data about the instances in the EC2 Auto Scaling group.

        :return: A list of instance IDs in the Auto Scaling group.
        """
        try:
            as_response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[self.group_name]
            )
            instance_ids = [
```

```
                i["InstanceId"]
                for i in as_response["AutoScalingGroups"][0]["Instances"]
            ]
            log.info(
                f"Retrieved {len(instance_ids)} instances for Auto Scaling group
{self.group_name}."
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to retrieve instances for Auto Scaling group
{self.group_name}."
            )
            if error_code == "ResourceNotFound":
                log.error(f"The Auto Scaling group '{self.group_name}' does not
exist.")
            log.error(f"Full error:\n\t{err}")
        else:
            return instance_ids


    def terminate_instance(self, instance_id: str, decrementsetting=False) ->
None:
        """
        Terminates an instance in an EC2 Auto Scaling group. After an instance is
        terminated, it can no longer be accessed.

        :param instance_id: The ID of the instance to terminate.
        :param decrementsetting: If True, do not replace terminated instances.
        """
        try:
            self.autoscaling_client.terminate_instance_in_auto_scaling_group(
                InstanceId=instance_id,
                ShouldDecrementDesiredCapacity=decrementsetting,
            )
            log.info("Terminated instance %s.", instance_id)

            # Adding a waiter to ensure the instance is terminated
            waiter = self.ec2_client.get_waiter("instance_terminated")
            log.info("Waiting for instance %s to be terminated...", instance_id)
            waiter.wait(InstanceIds=[instance_id])
            log.info(
                f"Instance '{instance_id}' has been terminated and will be
replaced."
```

```
            )

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to terminate instance '{instance_id}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to terminate the instance again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the resource."
                )
            log.error(f"Full error:\n\t{err}")

    def attach_load_balancer_target_group(
        self, lb_target_group: Dict[str, Any]
    ) -> None:
        """
        Attaches an Elastic Load Balancing (ELB) target group to this EC2 Auto
Scaling group.
        The target group specifies how the load balancer forwards requests to the
instances
        in the group.

        :param lb_target_group: Data about the ELB target group to attach.
        """
        try:
            self.autoscaling_client.attach_load_balancer_target_groups(
                AutoScalingGroupName=self.group_name,
                TargetGroupARNs=[lb_target_group["TargetGroupArn"]],
            )
            log.info(
                "Attached load balancer target group %s to auto scaling group
%s.",
                lb_target_group["TargetGroupName"],
                self.group_name,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
```

```
            log.error(
                f"Failed to attach load balancer target group
 '{lb_target_group['TargetGroupName']}'."
            )
            if error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
 the resource."
                )
            elif error_code == "ServiceLinkedRoleFailure":
                log.error(
                    "The operation failed because the service-linked role is not
 ready or does not exist. "
                    "Check that the service-linked role exists and is correctly
 configured."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_autoscaling_group(self, group_name: str) -> None:
        """
        Terminates all instances in the group, then deletes the EC2 Auto Scaling
 group.

        :param group_name: The name of the group to delete.
        """
        try:
            response = self.autoscaling_client.describe_auto_scaling_groups(
                AutoScalingGroupNames=[group_name]
            )
            groups = response.get("AutoScalingGroups", [])
            if len(groups) > 0:
                self.autoscaling_client.update_auto_scaling_group(
                    AutoScalingGroupName=group_name, MinSize=0
                )
                instance_ids = [inst["InstanceId"] for inst in groups[0]
["Instances"]]
                for inst_id in instance_ids:
                    self.terminate_instance(inst_id)

                # Wait for all instances to be terminated
                if instance_ids:
                    waiter = self.ec2_client.get_waiter("instance_terminated")
```

```
                    log.info("Waiting for all instances to be terminated...")
                    waiter.wait(InstanceIds=instance_ids)
                    log.info("All instances have been terminated.")
            else:
                log.info(f"No groups found named '{group_name}'! Nothing to do.")
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete Auto Scaling group '{group_name}'.")
            if error_code == "ScalingActivityInProgressFault":
                log.error(
                    "Scaling activity is currently in progress. "
                    "Wait for the scaling activity to complete before attempting
to delete the group again."
                )
            elif error_code == "ResourceContentionFault":
                log.error(
                    "The request failed due to a resource contention issue. "
                    "Ensure that no conflicting operations are being performed on
the group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_default_vpc(self) -> Dict[str, Any]:
        """
        Gets the default VPC for the account.

        :return: Data about the default VPC.
        """
        try:
            response = self.ec2_client.describe_vpcs(
                Filters=[{"Name": "is-default", "Values": ["true"]}]
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error("Failed to retrieve the default VPC.")
            if error_code == "UnauthorizedOperation":
                log.error(
                    "You do not have the necessary permissions to describe VPCs.
"
                    "Ensure that your AWS IAM user or role has the correct
permissions."
                )
            elif error_code == "InvalidParameterValue":
```

```
                log.error(
                    "One or more parameters are invalid. Check the request
 parameters."
                )

            log.error(f"Full error:\n\t{err}")
        else:
            if "Vpcs" in response and response["Vpcs"]:
                log.info(f"Retrieved default VPC: {response['Vpcs'][0]
['VpcId']}")
                return response["Vpcs"][0]
            else:
                pass


    def verify_inbound_port(
        self, vpc: Dict[str, Any], port: int, ip_address: str
    ) -> Tuple[Dict[str, Any], bool]:
        """
        Verify the default security group of the specified VPC allows ingress
 from this
        computer. This can be done by allowing ingress from this computer's IP
        address. In some situations, such as connecting from a corporate network,
 you
        must instead specify a prefix list ID. You can also temporarily open the
 port to
        any IP address while running this example. If you do, be sure to remove
 public
        access when you're done.

        :param vpc: The VPC used by this example.
        :param port: The port to verify.
        :param ip_address: This computer's IP address.
        :return: The default security group of the specified VPC, and a value
 that indicates
                 whether the specified port is open.
        """
        try:
            response = self.ec2_client.describe_security_groups(
                Filters=[
                    {"Name": "group-name", "Values": ["default"]},
                    {"Name": "vpc-id", "Values": [vpc["VpcId"]]},
                ]
            )
```

```
            sec_group = response["SecurityGroups"][0]
            port_is_open = False
            log.info(f"Found default security group {sec_group['GroupId']}.")

            for ip_perm in sec_group["IpPermissions"]:
                if ip_perm.get("FromPort", 0) == port:
                    log.info(f"Found inbound rule: {ip_perm}")
                    for ip_range in ip_perm["IpRanges"]:
                        cidr = ip_range.get("CidrIp", "")
                        if cidr.startswith(ip_address) or cidr == "0.0.0.0/0":
                            port_is_open = True
                    if ip_perm["PrefixListIds"]:
                        port_is_open = True
                    if not port_is_open:
                        log.info(
                            f"The inbound rule does not appear to be open to
either this computer's IP "
                            f"address of {ip_address}, to all IP addresses
(0.0.0.0/0), or to a prefix list ID."
                        )
                    else:
                        break
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to verify inbound rule for port {port} for VPC
{vpc['VpcId']}."
            )
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    f"The specified VPC ID '{vpc['VpcId']}' does not exist.
Please check the VPC ID."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return sec_group, port_is_open


    def open_inbound_port(self, sec_group_id: str, port: int, ip_address: str) ->
None:
        """
        Add an ingress rule to the specified security group that allows access on
the
        specified port from the specified IP address.
```

```
        :param sec_group_id: The ID of the security group to modify.
        :param port: The port to open.
        :param ip_address: The IP address that is granted access.
        """
        try:
            self.ec2_client.authorize_security_group_ingress(
                GroupId=sec_group_id,
                CidrIp=f"{ip_address}/32",
                FromPort=port,
                ToPort=port,
                IpProtocol="tcp",
            )
            log.info(
                "Authorized ingress to %s on port %s from %s.",
                sec_group_id,
                port,
                ip_address,
            )
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to authorize ingress to security group '{sec_group_id}'
on port {port} from {ip_address}."
            )
            if error_code == "InvalidGroupId.Malformed":
                log.error(
                    "The security group ID is malformed. "
                    "Please verify that the security group ID is correct."
                )
            elif error_code == "InvalidPermission.Duplicate":
                log.error(
                    "The specified rule already exists in the security group. "
                    "Check the existing rules for this security group."
                )
            log.error(f"Full error:\n\t{err}")


    def get_subnets(self, vpc_id: str, zones: List[str] = None) -> List[Dict[str,
Any]]:
        """
        Gets the default subnets in a VPC for a specified list of Availability
Zones.
```

```
        :param vpc_id: The ID of the VPC to look up.
        :param zones: The list of Availability Zones to look up.
        :return: The list of subnets found.
        """
        # Ensure that 'zones' is a list, even if None is passed
        if zones is None:
            zones = []
        try:
            paginator = self.ec2_client.get_paginator("describe_subnets")
            page_iterator = paginator.paginate(
                Filters=[
                    {"Name": "vpc-id", "Values": [vpc_id]},
                    {"Name": "availability-zone", "Values": zones},
                    {"Name": "default-for-az", "Values": ["true"]},
                ]
            )

            subnets = []
            for page in page_iterator:
                subnets.extend(page["Subnets"])

            log.info("Found %s subnets for the specified zones.", len(subnets))
            return subnets
        except ClientError as err:
            log.error(
                f"Failed to retrieve subnets for VPC '{vpc_id}' in zones
{zones}."
            )
            error_code = err.response["Error"]["Code"]
            if error_code == "InvalidVpcID.NotFound":
                log.error(
                    "The specified VPC ID does not exist. "
                    "Please check the VPC ID and try again."
                )
            # Add more error-specific handling as needed
            log.error(f"Full error:\n\t{err}")
```

创建一个包含弹性负载均衡操作的类。

```
class ElasticLoadBalancerWrapper:
    """Encapsulates Elastic Load Balancing (ELB) actions."""

    def __init__(self, elb_client: boto3.client):
        """
        Initializes the LoadBalancer class with the necessary parameters.
        """
        self.elb_client = elb_client


    def create_target_group(
        self, target_group_name: str, protocol: str, port: int, vpc_id: str
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing target group. The target group
 specifies how
        the load balancer forwards requests to instances in the group and how
 instance
        health is checked.

        To speed up this demo, the health check is configured with shortened
 times and
        lower thresholds. In production, you might want to decrease the
 sensitivity of
        your health checks to avoid unwanted failures.

        :param target_group_name: The name of the target group to create.
        :param protocol: The protocol to use to forward requests, such as 'HTTP'.
        :param port: The port to use to forward requests, such as 80.
        :param vpc_id: The ID of the VPC in which the load balancer exists.
        :return: Data about the newly created target group.
        """
        try:
            response = self.elb_client.create_target_group(
                Name=target_group_name,
                Protocol=protocol,
                Port=port,
                HealthCheckPath="/healthcheck",
                HealthCheckIntervalSeconds=10,
                HealthCheckTimeoutSeconds=5,
                HealthyThresholdCount=2,
                UnhealthyThresholdCount=2,
                VpcId=vpc_id,
```

```
            )
            target_group = response["TargetGroups"][0]
            log.info(f"Created load balancing target group
'{target_group_name}'.")
            return target_group
        except ClientError as err:
            log.error(
                f"Couldn't create load balancing target group
'{target_group_name}'."
            )
            error_code = err.response["Error"]["Code"]

            if error_code == "DuplicateTargetGroupName":
                log.error(
                    f"Target group name {target_group_name} already exists. "
                    "Check if the target group already exists."
                    "Consider using a different name or deleting the existing
target group if appropriate."
                )
            elif error_code == "TooManyTargetGroups":
                log.error(
                    "Too many target groups exist in the account. "
                    "Consider deleting unused target groups to create space for
new ones."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_target_group(self, target_group_name) -> None:
        """
        Deletes the target group.
        """
        try:
            # Describe the target group to get its ARN
            response =
self.elb_client.describe_target_groups(Names=[target_group_name])
            tg_arn = response["TargetGroups"][0]["TargetGroupArn"]

            # Delete the target group
            self.elb_client.delete_target_group(TargetGroupArn=tg_arn)
            log.info("Deleted load balancing target group %s.",
target_group_name)
```

```
            # Use a custom waiter to wait until the target group is no longer
available
            self.wait_for_target_group_deletion(self.elb_client, tg_arn)
            log.info("Target group %s successfully deleted.", target_group_name)

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to delete target group '{target_group_name}'.")
            if error_code == "TargetGroupNotFound":
                log.error(
                    "Load balancer target group either already deleted or never
existed. "
                    "Verify the name and check that the resource exists in the
AWS Console."
                )
            elif error_code == "ResourceInUseException":
                log.error(
                    "Target group still in use by another resource. "
                    "Ensure that the target group is no longer associated with
any load balancers or resources.",
                )
            log.error(f"Full error:\n\t{err}")

    def wait_for_target_group_deletion(
        self, elb_client, target_group_arn, max_attempts=10, delay=30
    ):
        for attempt in range(max_attempts):
            try:

elb_client.describe_target_groups(TargetGroupArns=[target_group_arn])
                print(
                    f"Attempt {attempt + 1}: Target group {target_group_arn}
still exists."
                )
            except ClientError as e:
                if e.response["Error"]["Code"] == "TargetGroupNotFound":
                    print(
                        f"Target group {target_group_arn} has been successfully
deleted."
                    )
                    return
                else:
                    raise
            time.sleep(delay)
```

```
        raise TimeoutError(
            f"Target group {target_group_arn} was not deleted after {max_attempts
* delay} seconds."
        )


    def create_load_balancer(
        self,
        load_balancer_name: str,
        subnet_ids: List[str],
    ) -> Dict[str, Any]:
        """
        Creates an Elastic Load Balancing load balancer that uses the specified
subnets
        and forwards requests to the specified target group.

        :param load_balancer_name: The name of the load balancer to create.
        :param subnet_ids: A list of subnets to associate with the load balancer.
        :return: Data about the newly created load balancer.
        """
        try:
            response = self.elb_client.create_load_balancer(
                Name=load_balancer_name, Subnets=subnet_ids
            )
            load_balancer = response["LoadBalancers"][0]
            log.info(f"Created load balancer '{load_balancer_name}'.")

            waiter = self.elb_client.get_waiter("load_balancer_available")
            log.info(
                f"Waiting for load balancer '{load_balancer_name}' to be
available..."
            )
            waiter.wait(Names=[load_balancer_name])
            log.info(f"Load balancer '{load_balancer_name}' is now available!")

        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to create load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "DuplicateLoadBalancerNameException":
                log.error(
```

```
                    f"A load balancer with the name '{load_balancer_name}'
already exists. "
                    "Load balancer names must be unique within the AWS region. "
                    "Please choose a different name and try again."
                )
            if error_code == "TooManyLoadBalancersException":
                log.error(
                    "The maximum number of load balancers has been reached in
this account and region. "
                    "You can delete unused load balancers or request an increase
in the service quota from AWS Support."
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return load_balancer


    def create_listener(
        self,
        load_balancer_name: str,
        target_group: Dict[str, Any],
    ) -> Dict[str, Any]:
        """
        Creates a listener for the specified load balancer that forwards requests
to the
        specified target group.

        :param load_balancer_name: The name of the load balancer to create a
listener for.
        :param target_group: An existing target group that is added as a listener
to the
                             load balancer.
        :return: Data about the newly created listener.
        """
        try:
            # Retrieve the load balancer ARN
            load_balancer_response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            load_balancer_arn = load_balancer_response["LoadBalancers"][0][
                "LoadBalancerArn"
            ]

            # Create the listener
```

```
            response = self.elb_client.create_listener(
                LoadBalancerArn=load_balancer_arn,
                Protocol=target_group["Protocol"],
                Port=target_group["Port"],
                DefaultActions=[
                    {
                        "Type": "forward",
                        "TargetGroupArn": target_group["TargetGroupArn"],
                    }
                ],
            )
            log.info(
                f"Created listener to forward traffic from load balancer
'{load_balancer_name}' to target group '{target_group['TargetGroupName']}'."
            )
            return response["Listeners"][0]
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Failed to add a listener on '{load_balancer_name}' for target
group '{target_group['TargetGroupName']}'."
            )

            if error_code == "ListenerNotFoundException":
                log.error(
                    f"The listener could not be found for the load balancer
'{load_balancer_name}'. "
                    "Please check the load balancer name and target group
configuration."
                )
            if error_code == "InvalidConfigurationRequestException":
                log.error(
                    f"The configuration provided for the listener on load
balancer '{load_balancer_name}' is invalid. "
                    "Please review the provided protocol, port, and target group
settings."
                )
            log.error(f"Full error:\n\t{err}")


    def delete_load_balancer(self, load_balancer_name) -> None:
        """
        Deletes a load balancer.
```

```
        :param load_balancer_name: The name of the load balancer to delete.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            lb_arn = response["LoadBalancers"][0]["LoadBalancerArn"]
            self.elb_client.delete_load_balancer(LoadBalancerArn=lb_arn)
            log.info("Deleted load balancer %s.", load_balancer_name)
            waiter = self.elb_client.get_waiter("load_balancers_deleted")
            log.info("Waiting for load balancer to be deleted...")
            waiter.wait(Names=[load_balancer_name])
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(
                f"Couldn't delete load balancer '{load_balancer_name}'. Error
code: {error_code}, Message: {err.response['Error']['Message']}"
            )

            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    f"The load balancer '{load_balancer_name}' does not exist. "
                    "Please check the name and try again."
                )
            log.error(f"Full error:\n\t{err}")


    def get_endpoint(self, load_balancer_name) -> str:
        """
        Gets the HTTP endpoint of the load balancer.

        :return: The endpoint.
        """
        try:
            response = self.elb_client.describe_load_balancers(
                Names=[load_balancer_name]
            )
            return response["LoadBalancers"][0]["DNSName"]
        except ClientError as err:
            log.error(
                f"Couldn't get the endpoint for load balancer
{load_balancer_name}"
            )
            error_code = err.response["Error"]["Code"]
```

```
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Verify load balancer name and ensure it exists in the AWS
console."
                )
            log.error(f"Full error:\n\t{err}")

    @staticmethod
    def verify_load_balancer_endpoint(endpoint) -> bool:
        """
        Verify this computer can successfully send a GET request to the load
balancer endpoint.

        :param endpoint: The endpoint to verify.
        :return: True if the GET request is successful, False otherwise.
        """
        retries = 3
        verified = False
        while not verified and retries > 0:
            try:
                lb_response = requests.get(f"http://{endpoint}")
                log.info(
                    "Got response %s from load balancer endpoint.",
                    lb_response.status_code,
                )
                if lb_response.status_code == 200:
                    verified = True
                else:
                    retries = 0
            except requests.exceptions.ConnectionError:
                log.info(
                    "Got connection error from load balancer endpoint,
retrying..."
                )
                retries -= 1
                time.sleep(10)
        return verified

    def check_target_health(self, target_group_name: str) -> List[Dict[str,
Any]]:
        """
        Checks the health of the instances in the target group.

        :return: The health status of the target group.
```

```
        """
        try:
            tg_response = self.elb_client.describe_target_groups(
                Names=[target_group_name]
            )
            health_response = self.elb_client.describe_target_health(
                TargetGroupArn=tg_response["TargetGroups"][0]["TargetGroupArn"]
            )
        except ClientError as err:
            log.error(f"Couldn't check health of {target_group_name} target(s).")
            error_code = err.response["Error"]["Code"]
            if error_code == "LoadBalancerNotFoundException":
                log.error(
                    "Load balancer associated with the target group was not
found. "
                    "Ensure the load balancer exists, is in the correct AWS
region, and "
                    "that you have the necessary permissions to access it.",
                )
            elif error_code == "TargetGroupNotFoundException":
                log.error(
                    "Target group was not found. "
                    "Verify the target group name, check that it exists in the
correct region, "
                    "and ensure it has not been deleted or created in a different
account.",
                )
            log.error(f"Full error:\n\t{err}")
        else:
            return health_response["TargetHealthDescriptions"]
```

创建一个使用 DynamoDB 模拟推荐服务的类。

```
class RecommendationService:
    """
    Encapsulates a DynamoDB table to use as a service that recommends books,
 movies,
    and songs.
    """
```

```
   def __init__(self, table_name: str, dynamodb_client: boto3.client):
       """
       Initializes the RecommendationService class with the necessary
parameters.

       :param table_name: The name of the DynamoDB recommendations table.
       :param dynamodb_client: A Boto3 DynamoDB client.
       """
       self.table_name = table_name
       self.dynamodb_client = dynamodb_client

   def create(self) -> Dict[str, Any]:
       """
       Creates a DynamoDB table to use as a recommendation service. The table
has a
       hash key named 'MediaType' that defines the type of media recommended,
such as
       Book or Movie, and a range key named 'ItemId' that, combined with the
MediaType,
       forms a unique identifier for the recommended item.

       :return: Data about the newly created table.
       :raises RecommendationServiceError: If the table creation fails.
       """
       try:
           response = self.dynamodb_client.create_table(
               TableName=self.table_name,
               AttributeDefinitions=[
                   {"AttributeName": "MediaType", "AttributeType": "S"},
                   {"AttributeName": "ItemId", "AttributeType": "N"},
               ],
               KeySchema=[
                   {"AttributeName": "MediaType", "KeyType": "HASH"},
                   {"AttributeName": "ItemId", "KeyType": "RANGE"},
               ],
               ProvisionedThroughput={"ReadCapacityUnits": 5,
"WriteCapacityUnits": 5},
           )
           log.info("Creating table %s...", self.table_name)
           waiter = self.dynamodb_client.get_waiter("table_exists")
           waiter.wait(TableName=self.table_name)
           log.info("Table %s created.", self.table_name)
       except ClientError as err:
```

```
            if err.response["Error"]["Code"] == "ResourceInUseException":
                log.info("Table %s exists, nothing to be done.", self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when creating table: {err}."
                )
        else:
            return response

    def populate(self, data_file: str) -> None:
        """
        Populates the recommendations table from a JSON file.

        :param data_file: The path to the data file.
        :raises RecommendationServiceError: If the table population fails.
        """
        try:
            with open(data_file) as data:
                items = json.load(data)
            batch = [{"PutRequest": {"Item": item}} for item in items]
            self.dynamodb_client.batch_write_item(RequestItems={self.table_name:
batch})
            log.info(
                "Populated table %s with items from %s.", self.table_name,
data_file
            )
        except ClientError as err:
            raise RecommendationServiceError(
                self.table_name, f"Couldn't populate table from {data_file}:
{err}"
            )

    def destroy(self) -> None:
        """
        Deletes the recommendations table.

        :raises RecommendationServiceError: If the table deletion fails.
        """
        try:
            self.dynamodb_client.delete_table(TableName=self.table_name)
            log.info("Deleting table %s...", self.table_name)
            waiter = self.dynamodb_client.get_waiter("table_not_exists")
            waiter.wait(TableName=self.table_name)
            log.info("Table %s deleted.", self.table_name)
```

```
        except ClientError as err:
            if err.response["Error"]["Code"] == "ResourceNotFoundException":
                log.info("Table %s does not exist, nothing to do.",
 self.table_name)
            else:
                raise RecommendationServiceError(
                    self.table_name, f"ClientError when deleting table: {err}."
                )
```

创建一个包含 Systems Manager 操作的类。

```
class ParameterHelper:
    """
    Encapsulates Systems Manager parameters. This example uses these parameters
 to drive
    the demonstration of resilient architecture, such as failure of a dependency
 or
    how the service responds to a health check.
    """

    table: str = "doc-example-resilient-architecture-table"
    failure_response: str = "doc-example-resilient-architecture-failure-response"
    health_check: str = "doc-example-resilient-architecture-health-check"

    def __init__(self, table_name: str, ssm_client: boto3.client):
        """
        Initializes the ParameterHelper class with the necessary parameters.

        :param table_name: The name of the DynamoDB table that is used as a
 recommendation
                           service.
        :param ssm_client: A Boto3 Systems Manager client.
        """
        self.ssm_client = ssm_client
        self.table_name = table_name

    def reset(self) -> None:
        """
        Resets the Systems Manager parameters to starting values for the demo.
```

```
        These are the name of the DynamoDB recommendation table, no response when
a
        dependency fails, and shallow health checks.
        """
        self.put(self.table, self.table_name)
        self.put(self.failure_response, "none")
        self.put(self.health_check, "shallow")

    def put(self, name: str, value: str) -> None:
        """
        Sets the value of a named Systems Manager parameter.

        :param name: The name of the parameter.
        :param value: The new value of the parameter.
        :raises ParameterHelperError: If the parameter value cannot be set.
        """
        try:
            self.ssm_client.put_parameter(
                Name=name, Value=value, Overwrite=True, Type="String"
            )
            log.info("Setting parameter %s to '%s'.", name, value)
        except ClientError as err:
            error_code = err.response["Error"]["Code"]
            log.error(f"Failed to set parameter {name}.")
            if error_code == "ParameterLimitExceeded":
                log.error(
                    "The parameter limit has been exceeded. "
                    "Consider deleting unused parameters or request a limit
increase."
                )
            elif error_code == "ParameterAlreadyExists":
                log.error(
                    "The parameter already exists and overwrite is set to False.
"
                    "Use Overwrite=True to update the parameter."
                )
            log.error(f"Full error:\n\t{err}")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。

- AttachLoadBalancerTargetGroups
- CreateAutoScalingGroup
- CreateInstanceProfile
- CreateLaunchTemplate
- CreateListener
- CreateLoadBalancer
- CreateTargetGroup
- DeleteAutoScalingGroup
- DeleteInstanceProfile
- DeleteLaunchTemplate
- DeleteLoadBalancer
- DeleteTargetGroup
- DescribeAutoScalingGroups
- DescribeAvailabilityZones
- DescribeIamInstanceProfileAssociations
- DescribeInstances
- DescribeLoadBalancers
- DescribeSubnets
- DescribeTargetGroups
- DescribeTargetHealth
- DescribeVpcs
- RebootInstances
- ReplaceIamInstanceProfileAssociation
- TerminateInstanceInAutoScalingGroup
- UpdateAutoScalingGroup

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

配置 Amazon ECS Service Connect

以下代码示例展示了如何：

- 创建 VPC 基础设施
- 设置日志记录
- 创建 ECS 集群
- 配置 IAM 角色
- 使用 Service Connect 创建服务
- 验证部署
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
```

```
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}
```

```
# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
```

```
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
```

```
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
```

```
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
```

```
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
```

```
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
```

```
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
```

```
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
```

```
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi
```

```
    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
    echo "CREATED RESOURCES"
    echo "============================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"
```

```
        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
```

```
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json

    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
```

```
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================"
    echo "CLEANUP CONFIRMATION"
    echo "============================================"
    echo "Do you want to clean up all created resources? (y/n): "
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
```

```
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters
  - DescribeServices
  - DescribeSubnets
  - DescribeTasks
  - DescribeVpcs

- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Lambda 代理集成创建 REST API

以下代码示例展示了如何：

- 为 Lambda 执行创建 IAM 角色
- 创建并部署 Lambda 函数
- 创建 REST API
- 配置 Lambda 代理集成
- 部署并测试 API
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi

echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
```

```
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
```

```
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration
```

```
# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
```

```
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
```

```
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
```

```
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddPermission
  - AttachRolePolicy

- CreateDeployment
- CreateFunction
- CreateResource
- CreateRestApi
- CreateRole
- DeleteFunction
- DeleteRestApi
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetResources
- GetRestApis
- PutIntegration
- PutMethod

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Fargate 启动类型的 Amazon ECS Linux 任务

以下代码示例展示了如何：

- 创建集群
- 创建任务定义
- 创建服务
- 清理

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"
```

```
    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}

# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
```

```
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."

        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
```

```
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"

        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
```

```
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
 then
            echo ""
            echo "Step 1: Scaling service to 0 tasks..."
            if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
                echo "Waiting for service to stabilize after scaling to 0..."
                execute_command "aws ecs wait services-stable --cluster
 '$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

                echo "Deleting service..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
 --service '$SERVICE_NAME'" "Delete ECS service"
            else
```

```
                echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
                execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
            fi
        fi

        # Step 2: Wait a bit for tasks to fully terminate
        echo ""
        echo "Step 2: Waiting for tasks to fully terminate..."
        sleep 15

        # Step 3: Delete cluster
        if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
            echo ""
            echo "Step 3: Deleting cluster..."
            execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
        fi

        # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
        if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
            echo ""
            echo "Step 4: Cleaning up security group..."

            # First, wait for network interfaces to be cleaned up
            wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"

            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
" ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
```

```
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "==========================================="
        echo "CLEANUP COMPLETED"
        echo "==========================================="
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}
```

```
# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "==========================================="
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "==========================================="

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================="
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================="

CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================="
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================="

# Create task definition JSON with validated inputs
```

```
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
```

```
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================="
echo "STEP 4: SET UP NETWORKING"
echo "============================================="

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
```

```
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
```

```
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "============================================="
echo "STEP 5: CREATE ECS SERVICE"
echo "============================================="

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "============================================="
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "============================================="

echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"
```

```
# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
    else
        echo ""
        echo "========================================="
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "========================================="
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "========================================="
```

```
echo "SERVICE INFORMATION"
echo "============================================"
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "============================================"
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "============================================"
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeNetworkInterfaces
  - DescribeSecurityGroups
  - DescribeServices
  - DescribeSubnets

- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包创建只读和读写 IAM 用户

以下代码示例显示如何创建用户并将策略附加到用户。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建两个 IAM 用户。
- 附加一项策略，让一个用户获取对象并将其放入 Amazon S3 存储桶中。
- 附加一项策略，让另一个用户从该存储桶获取对象。
- 根据用户凭证获取对存储桶的不同权限。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 用户操作的函数。

```
import logging
import time

import boto3
from botocore.exceptions import ClientError

import access_key_wrapper
import policy_wrapper

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_user(user_name):
    """
    Creates a user. By default, a user has no permissions or access keys.

    :param user_name: The name of the user.
    :return: The newly created user.
    """
    try:
        user = iam.create_user(UserName=user_name)
        logger.info("Created user %s.", user.name)
    except ClientError:
        logger.exception("Couldn't create user %s.", user_name)
        raise
    else:
        return user



def update_user(user_name, new_user_name):
```

```
    """
    Updates a user's name.

    :param user_name: The current name of the user to update.
    :param new_user_name: The new name to assign to the user.
    :return: The updated user.
    """
    try:
        user = iam.User(user_name)
        user.update(NewUserName=new_user_name)
        logger.info("Renamed %s to %s.", user_name, new_user_name)
    except ClientError:
        logger.exception("Couldn't update name for user %s.", user_name)
        raise
    return user



def list_users():
    """
    Lists the users in the current account.

    :return: The list of users.
    """
    try:
        users = list(iam.users.all())
        logger.info("Got %s users.", len(users))
    except ClientError:
        logger.exception("Couldn't get users.")
        raise
    else:
        return users



def delete_user(user_name):
    """
    Deletes a user. Before a user can be deleted, all associated resources,
    such as access keys and policies, must be deleted or detached.

    :param user_name: The name of the user.
    """
    try:
        iam.User(user_name).delete()
```

```
        logger.info("Deleted user %s.", user_name)
    except ClientError:
        logger.exception("Couldn't delete user %s.", user_name)
        raise



def attach_policy(user_name, policy_arn):
    """
    Attaches a policy to a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to user %s.", policy_arn,
 user_name)
        raise



def detach_policy(user_name, policy_arn):
    """
    Detaches a policy from a user.

    :param user_name: The name of the user.
    :param policy_arn: The Amazon Resource Name (ARN) of the policy.
    """
    try:
        iam.User(user_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from user %s.", policy_arn, user_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from user %s.", policy_arn, user_name
        )
        raise
```

创建包装 IAM policy 操作的函数。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
```

```
        raise
    else:
        return policy



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

创建包装 IAM 访问密钥操作的函数。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.

    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
```

```
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

使用包装器函数创建具有不同策略的用户，然后使用其凭证访问 Amazon S3 存储桶。

```
def usage_demo():
    """
    Shows how to manage users, keys, and policies.
    This demonstration creates two users: one user who can put and get objects in
 an
    Amazon S3 bucket, and another user who can only get objects from the bucket.
    The demo then shows how the users can perform only the actions they are
 permitted
    to perform.
    """
```

```
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management user demo.")
    print("-" * 88)
    print(
        "Users can have policies and roles attached to grant them specific "
        "permissions."
    )
    s3 = boto3.resource("s3")
    bucket = s3.create_bucket(
        Bucket=f"demo-iam-bucket-{time.time_ns()}",
        CreateBucketConfiguration={
            "LocationConstraint": s3.meta.client.meta.region_name
        },
    )
    print(f"Created an Amazon S3 bucket named {bucket.name}.")
    user_read_writer = create_user("demo-iam-read-writer")
    user_reader = create_user("demo-iam-reader")
    print(f"Created two IAM users: {user_read_writer.name} and
{user_reader.name}")
    update_user(user_read_writer.name, "demo-iam-creator")
    update_user(user_reader.name, "demo-iam-getter")
    users = list_users()
    user_read_writer = next(
        user for user in users if user.user_id == user_read_writer.user_id
    )
    user_reader = next(user for user in users if user.user_id ==
user_reader.user_id)
    print(
        f"Changed the names of the users to {user_read_writer.name} "
        f"and {user_reader.name}."
    )

    read_write_policy = policy_wrapper.create_policy(
        "demo-iam-read-write-policy",
        "Grants rights to create and get an object in the demo bucket.",
        ["s3:PutObject", "s3:GetObject"],
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(
        f"Created policy {read_write_policy.policy_name} with ARN:
{read_write_policy.arn}"
    )
    print(read_write_policy.description)
```

```
    read_policy = policy_wrapper.create_policy(
        "demo-iam-read-policy",
        "Grants rights to get an object from the demo bucket.",
        "s3:GetObject",
        f"arn:aws:s3:::{bucket.name}/*",
    )
    print(f"Created policy {read_policy.policy_name} with ARN:
{read_policy.arn}")
    print(read_policy.description)
    attach_policy(user_read_writer.name, read_write_policy.arn)
    print(f"Attached {read_write_policy.policy_name} to
{user_read_writer.name}.")
    attach_policy(user_reader.name, read_policy.arn)
    print(f"Attached {read_policy.policy_name} to {user_reader.name}.")

    user_read_writer_key = access_key_wrapper.create_key(user_read_writer.name)
    print(f"Created access key pair for {user_read_writer.name}.")
    user_reader_key = access_key_wrapper.create_key(user_reader.name)
    print(f"Created access key pair for {user_reader.name}.")

    s3_read_writer_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_read_writer_key.id,
        aws_secret_access_key=user_read_writer_key.secret,
    )
    demo_object_key = f"object-{time.time_ns()}"
    demo_object = None
    while demo_object is None:
        try:
            demo_object = s3_read_writer_resource.Bucket(bucket.name).put_object(
                Key=demo_object_key, Body=b"AWS IAM demo object content!"
            )
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise
    print(
        f"Put {demo_object_key} into {bucket.name} using "
        f"{user_read_writer.name}'s credentials."
    )

    read_writer_object = s3_read_writer_resource.Bucket(bucket.name).Object(
```

```
        demo_object_key
    )
    read_writer_content = read_writer_object.get()["Body"].read()
    print(f"Got object {read_writer_object.key} using read-writer user's
credentials.")
    print(f"Object content: {read_writer_content}")

    s3_reader_resource = boto3.resource(
        "s3",
        aws_access_key_id=user_reader_key.id,
        aws_secret_access_key=user_reader_key.secret,
    )
    demo_content = None
    while demo_content is None:
        try:
            demo_object =
s3_reader_resource.Bucket(bucket.name).Object(demo_object_key)
            demo_content = demo_object.get()["Body"].read()
            print(f"Got object {demo_object.key} using reader user's
credentials.")
            print(f"Object content: {demo_content}")
        except ClientError as error:
            if error.response["Error"]["Code"] == "InvalidAccessKeyId":
                print("Access key not yet available. Waiting...")
                time.sleep(1)
            else:
                raise

    try:
        demo_object.delete()
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("-" * 88)
            print(
                "Tried to delete the object using the reader user's credentials.
"
                "Got expected AccessDenied error because the reader is not "
                "allowed to delete objects."
            )
            print("-" * 88)

    access_key_wrapper.delete_key(user_reader.name, user_reader_key.id)
    detach_policy(user_reader.name, read_policy.arn)
    policy_wrapper.delete_policy(read_policy.arn)
```

```
    delete_user(user_reader.name)
    print(f"Deleted keys, detached and deleted policy, and deleted
{user_reader.name}.")

    access_key_wrapper.delete_key(user_read_writer.name, user_read_writer_key.id)
    detach_policy(user_read_writer.name, read_write_policy.arn)
    policy_wrapper.delete_policy(read_write_policy.arn)
    delete_user(user_read_writer.name)
    print(
        f"Deleted keys, detached and deleted policy, and deleted
{user_read_writer.name}."
    )

    bucket.objects.delete()
    bucket.delete()
    print(f"Emptied and deleted {bucket.name}.")
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - AttachUserPolicy
  - CreateAccessKey
  - CreatePolicy
  - CreateUser
  - DeleteAccessKey
  - DeletePolicy
  - DeleteUser
  - DetachUserPolicy
  - ListUsers
  - UpdateUser

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将函数名称作为变量创建 CloudWatch 控制面板

以下代码示例展示了如何：

- 创建 CloudWatch 控制面板
- 添加带有函数名称变量的 Lambda 指标小组件
- 验证控制面板
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
```

```
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "=========================================="
    echo "CLEANUP CONFIRMATION"
    echo "=========================================="
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true

    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
```

```
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."

    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'
```

```
    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
```

```
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
```

```
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
```

```
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
```

```
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
```

```
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
```

```
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。

- AttachRolePolicy
- CreateFunction
- CreateRole
- DeleteDashboards
- DeleteFunction
- DeleteRole
- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 EC2 启动类型的 Amazon ECS 服务

以下代码示例展示了如何：

- 创建 ECS 集群
- 创建和监控服务
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"
```

```
    echo ""
    echo "============================================="
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "============================================="
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done

    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
```

```
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
 "$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
 $TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
 2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
```

```
   if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
       log "Deleting security group: $SECURITY_GROUP_ID"
       local retry_count=0
       local max_retries=3

       while [[ $retry_count -lt $max_retries ]]; do
           if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
               log "Successfully deleted security group"
               break
           else
               ((retry_count++))
               if [[ $retry_count -lt $max_retries ]]; then
                   log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                   sleep 10
               else
                   log "ERROR: Failed to delete security group after
$max_retries attempts"
                   ((cleanup_errors++))
               fi
           fi
       done
   fi

   # Delete key pair
   if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
       log "Deleting key pair: $KEY_PAIR_NAME"
       if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
           log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
           ((cleanup_errors++))
       fi
       rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
   fi

   # Delete ECS cluster
   if [[ -n "${CLUSTER_NAME:-}" ]]; then
       log "Deleting ECS cluster: $CLUSTER_NAME"
       if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
           log "WARNING: Failed to delete cluster $CLUSTER_NAME"
           ((cleanup_errors++))
```

```
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
 file for details."
    fi
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
```

```
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")
```

```
    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
```

```
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}

# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
```

```
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role

        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
```

```
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
```

```
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi
```

```
    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."

    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}
```

```
# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
```

```
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10

        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "========================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "========================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
```

```
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "=========================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "=========================================="
    echo "Resources created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "=========================================="
    echo "CLEANUP CONFIRMATION"
    echo "=========================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile

- CreateKeyPair
- CreateRole
- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteKeyPair
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Managed Grafana 工作区

以下代码示例展示了如何：

- 为工作区创建 IAM 角色
- 创建 Grafana 工作区
- 配置身份验证
- 配置可选设置
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
```

```
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"
```

```
# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
```

```
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
```

```
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"

# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
```

```
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "============================================="
echo "WORKSPACE INFORMATION"
echo "============================================="
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "============================================="

# Step 6: Prompt for cleanup
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
```

```
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"

    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"
```

```
    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建第一个 Lambda 函数

以下代码示例展示了如何：

- 为 Lambda 创建 IAM 角色
- 创建函数代码

- 创建 Lambda 函数
- 测试 Lambda 函数
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash
# AWS Lambda - Create Your First Function
# This script creates a Lambda function, invokes it with a test event,
# views CloudWatch logs, and cleans up all resources.
#
# Source: https://docs.aws.amazon.com/lambda/latest/dg/getting-started.html
#
# Resources created:
#   - IAM role (Lambda execution role with basic logging permissions)
#   - Lambda function (Python 3.13 or Node.js 22.x runtime)
#   - CloudWatch log group (created automatically by Lambda on invocation)

set -eE -o pipefail

################################################################################
# Setup
################################################################################

UNIQUE_ID=$(head -c 8 /dev/urandom | od -An -tx1 | tr -d ' ')
FUNCTION_NAME="my-lambda-function-${UNIQUE_ID}"
ROLE_NAME="lambda-execution-role-${UNIQUE_ID}"
LOG_GROUP_NAME="/aws/lambda/${FUNCTION_NAME}"

TEMP_DIR=$(mktemp -d)
readonly TEMP_DIR
LOG_FILE="${TEMP_DIR}/lambda-gettingstarted.log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

declare -a CREATED_RESOURCES

################################################################################
# Helper functions
################################################################################

cleanup_resources() {
    # Disable error trap to prevent recursion during cleanup
    trap - ERR
    set +eE

    echo ""
    echo "Cleaning up resources..."
    echo ""

    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        local RESOURCE="${CREATED_RESOURCES[$i]}"
        local TYPE="${RESOURCE%%:*}"
        local NAME="${RESOURCE#*:}"

        case "$TYPE" in
            log-group)
                echo "Deleting CloudWatch log group: ${NAME}"
                aws logs delete-log-group \
                    --log-group-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete log group ${NAME}."
                ;;
            lambda-function)
                echo "Deleting Lambda function: ${NAME}"
                aws lambda delete-function \
                    --function-name "$NAME" 2>&1 || echo "  WARNING: Could not
 delete Lambda function ${NAME}."
                echo "  Waiting for function deletion to complete..."
                local DELETE_WAIT=0
                while aws lambda get-function --function-name "$NAME" > /dev/null
 2>&1; do
                    sleep 2
                    DELETE_WAIT=$((DELETE_WAIT + 2))
                    if [ "$DELETE_WAIT" -ge 60 ]; then
                        echo "  WARNING: Timed out waiting for function
 deletion."
                        break
```

```
                    fi
                done
                ;;
            iam-role-policy)
                local ROLE_PART="${NAME%%|*}"
                local POLICY_PART="${NAME#*|}"
                echo "Detaching policy from role: ${ROLE_PART}"
                aws iam detach-role-policy \
                    --role-name "$ROLE_PART" \
                    --policy-arn "$POLICY_PART" 2>&1 || echo "  WARNING: Could
 not detach policy from role ${ROLE_PART}."
                ;;
            iam-role)
                echo "Deleting IAM role: ${NAME}"
                aws iam delete-role \
                    --role-name "$NAME" 2>&1 || echo "  WARNING: Could not delete
 IAM role ${NAME}."
                ;;
        esac
    done

    if [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi

    echo ""
    echo "Cleanup complete."
}

handle_error() {
    echo ""
    echo "========================================="
    echo "ERROR: Script failed at $1"
    echo "========================================="
    echo ""
    if [ ${#CREATED_RESOURCES[@]} -gt 0 ]; then
        echo "Attempting to clean up ${#CREATED_RESOURCES[@]} resource(s)..."
        cleanup_resources
    fi
    exit 1
}

trap 'handle_error "line $LINENO"' ERR
```

```
wait_for_resource() {
    local DESCRIPTION="$1"
    local COMMAND="$2"
    local TARGET_VALUE="$3"
    local TIMEOUT=300
    local ELAPSED=0
    local INTERVAL=5

    echo "Waiting for ${DESCRIPTION}..."
    while true; do
        local RESULT
        RESULT=$(eval "$COMMAND" 2>&1) || true
        if echo "$RESULT" | grep -q "$TARGET_VALUE"; then
            echo "  ${DESCRIPTION} is ready."
            return 0
        fi
        if [ "$ELAPSED" -ge "$TIMEOUT" ]; then
            echo "ERROR: Timed out waiting for ${DESCRIPTION} after ${TIMEOUT}
 seconds."
            return 1
        fi
        sleep "$INTERVAL"
        ELAPSED=$((ELAPSED + INTERVAL))
    done
}

validate_input() {
    local input="$1"
    local pattern="$2"
    if ! [[ "$input" =~ $pattern ]]; then
        echo "ERROR: Invalid input: $input"
        return 1
    fi
    return 0
}

################################################################################
# Region pre-check
################################################################################

CONFIGURED_REGION=$(aws configure get region 2>/dev/null || true)
if [ -z "$CONFIGURED_REGION" ] && [ -z "$AWS_DEFAULT_REGION" ] && [ -z
 "$AWS_REGION" ]; then
    echo "ERROR: No AWS region configured."
```

```
    echo "Run 'aws configure set region <region>' or export AWS_DEFAULT_REGION."
    exit 1
fi

################################################################################
# Runtime selection
################################################################################

echo ""
echo "=========================================="
echo "AWS Lambda - Create Your First Function"
echo "=========================================="
echo ""
echo "Select a runtime for your Lambda function:"
echo "  1) Python 3.13"
echo "  2) Node.js 22.x"
echo ""
echo "Using default: Python 3.13"
RUNTIME_CHOICE="1"

case "$RUNTIME_CHOICE" in
    1)
        RUNTIME="python3.13"
        HANDLER="lambda_function.lambda_handler"
        CODE_FILE="lambda_function.py"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'PYTHON_EOF'
import json
import logging
logger = logging.getLogger()
logger.setLevel(logging.INFO)
def lambda_handler(event, context):
    if not isinstance(event, dict) or 'length' not in event or 'width' not in
 event:
        raise ValueError('Event must contain length and width')
    try:
        length = float(event['length'])
        width = float(event['width'])
        if length < 0 or width < 0:
            raise ValueError('Length and width must be non-negative')
        area = calculate_area(length, width)
        print(f'The area is {area}')
        logger.info(f'CloudWatch logs group: {context.log_group_name}')
        return json.dumps({'area': area})
    except (TypeError, ValueError) as e:
```

```
        logger.error(f'Error processing input: {str(e)}')
        raise
def calculate_area(length, width):
    return length * width
PYTHON_EOF
        echo "Selected runtime: Python 3.13"
        ;;
    2)
        RUNTIME="nodejs22.x"
        HANDLER="index.handler"
        CODE_FILE="index.mjs"
        cat > "${TEMP_DIR}/${CODE_FILE}" << 'NODEJS_EOF'
export const handler = async (event, context) => {
  if (!event || typeof event.length !== 'number' || typeof event.width !==
 'number') {
    throw new Error('Event must contain numeric length and width');
  }
  if (event.length < 0 || event.width < 0) {
    throw new Error('Length and width must be non-negative');
  }
  const area = event.length * event.width;
  console.log(`The area is ${area}`);
  console.log('CloudWatch log group: ', context.logGroupName);
  return JSON.stringify({area});
};
NODEJS_EOF
        echo "Selected runtime: Node.js 22.x"
        ;;
    *)
        echo "ERROR: Invalid choice. Please enter 1 or 2."
        exit 1
        ;;
esac

# ============================================================================
# Step 1: Create IAM execution role
# ============================================================================

echo ""
echo "=========================================="
echo "Step 1: Create IAM execution role"
echo "=========================================="
echo ""
```

```
TRUST_POLICY='{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "lambda.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}'

echo "Creating IAM role: ${ROLE_NAME}"
ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" \
    --query 'Role.Arn' \
    --output text 2>&1)

if ! validate_input "$ROLE_OUTPUT" "^arn:aws:iam::[0-9]+:role/"; then
    echo "ERROR: Failed to create IAM role"
    exit 1
fi

echo "$ROLE_OUTPUT"
ROLE_ARN="$ROLE_OUTPUT"
CREATED_RESOURCES+=("iam-role:${ROLE_NAME}")
echo "Role ARN: ${ROLE_ARN}"

aws iam tag-role \
    --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=lambda-gettingstarted

echo ""
echo "Attaching AWSLambdaBasicExecutionRole policy..."
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>&1
CREATED_RESOURCES+=("iam-role-policy:${ROLE_NAME}|arn:aws:iam::aws:policy/
service-role/AWSLambdaBasicExecutionRole")
echo "Policy attached."
```

```
# IAM roles can take a few seconds to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

################################################################################
# Step 2: Create Lambda function
################################################################################

echo ""
echo "=========================================="
echo "Step 2: Create Lambda function"
echo "=========================================="
echo ""

echo "Creating deployment package..."
ORIGINAL_DIR=$(pwd)
cd "$TEMP_DIR" || exit 1
zip -j function.zip "$CODE_FILE" > /dev/null 2>&1 || {
    echo "ERROR: Failed to create deployment package"
    exit 1
}
cd "$ORIGINAL_DIR" || exit 1

if [ ! -f "${TEMP_DIR}/function.zip" ]; then
    echo "ERROR: Deployment package creation failed"
    exit 1
fi

echo "Creating Lambda function: ${FUNCTION_NAME}"
echo "  Runtime: ${RUNTIME}"
echo "  Handler: ${HANDLER}"
echo ""

CREATE_OUTPUT=$(aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime "$RUNTIME" \
    --role "$ROLE_ARN" \
    --handler "$HANDLER" \
    --architectures x86_64 \
    --zip-file "fileb://${TEMP_DIR}/function.zip" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted \
    --query '[FunctionName, FunctionArn, Runtime, State]' \
    --output text 2>&1)
```

```
if [ -z "$CREATE_OUTPUT" ]; then
    echo "ERROR: Failed to create Lambda function"
    exit 1
fi

echo "$CREATE_OUTPUT"
CREATED_RESOURCES+=("lambda-function:${FUNCTION_NAME}")

wait_for_resource "Lambda function to become Active" \
    "aws lambda get-function-configuration --function-name ${FUNCTION_NAME} --
query State --output text" \
    "Active"

################################################################################
# Step 3: Invoke the function
################################################################################

echo ""
echo "============================================"
echo "Step 3: Invoke the function"
echo "============================================"
echo ""

TEST_EVENT='{"length": 6, "width": 7}'
echo "Invoking function with test event: ${TEST_EVENT}"
echo ""

echo "$TEST_EVENT" > "${TEMP_DIR}/test-event.json"

if ! validate_input "$TEST_EVENT" '"length": [0-9]+, "width": [0-9]+'; then
    echo "ERROR: Invalid test event format"
    exit 1
fi

INVOKE_OUTPUT=$(aws lambda invoke \
    --function-name "$FUNCTION_NAME" \
    --payload "fileb://${TEMP_DIR}/test-event.json" \
    --cli-read-timeout 30 \
    "${TEMP_DIR}/response.json" 2>&1)
echo "$INVOKE_OUTPUT"

if [ ! -f "${TEMP_DIR}/response.json" ]; then
    echo "ERROR: No response file generated"
    exit 1
```

```
fi

RESPONSE=$(cat "${TEMP_DIR}/response.json")
echo ""
echo "Function response: ${RESPONSE}"
echo ""

if echo "$INVOKE_OUTPUT" | grep -qi "functionerror"; then
    echo "WARNING: Function returned an error."
fi

################################################################################
# Step 4: View CloudWatch logs
################################################################################

echo ""
echo "============================================"
echo "Step 4: View CloudWatch Logs"
echo "============================================"
echo ""

echo "Log group: ${LOG_GROUP_NAME}"
echo ""

echo "Waiting for CloudWatch logs to be available..."

LOG_STREAMS=""
for i in $(seq 1 6); do
    LOG_STREAMS=$(aws logs describe-log-streams \
        --log-group-name "$LOG_GROUP_NAME" \
        --order-by LastEventTime \
        --descending \
        --query 'logStreams[0].logStreamName' \
        --output text 2>/dev/null) || true
    if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
        break
    fi
    LOG_STREAMS=""
    sleep 5
done

if [ -n "$LOG_STREAMS" ] && [ "$LOG_STREAMS" != "None" ]; then
    echo "Latest log stream: ${LOG_STREAMS}"
    echo ""
```

```
    echo "--- Log events ---"
    LOG_EVENTS=$(aws logs get-log-events \
        --log-group-name "$LOG_GROUP_NAME" \
        --log-stream-name "$LOG_STREAMS" \
        --query 'events[].message' \
        --output text 2>&1) || true
    echo "$LOG_EVENTS"
    echo "--- End of log events ---"
else
    echo "No log streams found yet. Logs may take a moment to appear."
    echo "You can view them in the CloudWatch console:"
    echo "  Log group: ${LOG_GROUP_NAME}"
fi

CREATED_RESOURCES+=("log-group:${LOG_GROUP_NAME}")

aws logs tag-log-group \
    --log-group-name "$LOG_GROUP_NAME" \
    --tags project=doc-smith,tutorial=lambda-gettingstarted

################################################################################
# Summary and cleanup
################################################################################

echo ""
echo "==========================================="
echo "SUMMARY"
echo "==========================================="
echo ""
echo "Resources created:"
echo "  IAM role:          ${ROLE_NAME}"
echo "  Lambda function:   ${FUNCTION_NAME}"
echo "  CloudWatch logs:   ${LOG_GROUP_NAME}"
echo ""
echo "==========================================="
echo "CLEANUP"
echo "==========================================="
echo ""
echo "Cleaning up all created resources..."
cleanup_resources

echo ""
echo "Done."
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteFunction
  - DeleteLogGroup
  - DeleteRole
  - DescribeLogStreams
  - DetachRolePolicy
  - GetFunction
  - GetFunctionConfiguration
  - GetLogEvents
  - Invoke

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 CLI 开始使用 Redshift Serverless

以下代码示例展示了如何：

- 使用 secrets-manager CreateSecret
- 使用 secrets-manager DeleteSecret
- 使用 secrets-manager GetSecretValue
- 使用 redshift CreateNamespace
- 使用 redshift CreateWorkgroup
- 使用 redshift DeleteNamespace
- 使用 iam CreateRole

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Redshift Serverless Tutorial Script with Secrets Manager (No jq
 dependency)
# This script creates a Redshift Serverless environment, loads sample data, and
 runs queries
# Uses AWS Secrets Manager for secure password management without requiring jq

# Set up logging
LOG_FILE="redshift-serverless-tutorial-v4.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift Serverless tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
  local output=$1
  local cmd=$2

  if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Command failed: $cmd"
    echo "Output: $output"
    cleanup_resources
    exit 1
  fi
}

# Function to generate a secure password that meets Redshift requirements
generate_secure_password() {
  # Redshift password requirements:
  # - 8-64 characters
```

```
  # - At least one uppercase letter
  # - At least one lowercase letter
  # - At least one decimal digit
  # - Can contain printable ASCII characters except /, ", ', \, @, space

  local password=""
  local valid=false
  local attempts=0
  local max_attempts=10

  while [[ "$valid" == false && $attempts -lt $max_attempts ]]; do
    # Generate base password with safe characters
    local base=$(openssl rand -base64 12 | tr -d '/+=' | head -c 12)

    # Ensure we have at least one of each required character type
    local upper=$(echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" | fold -w1 | shuf -n1)
    local lower=$(echo "abcdefghijklmnopqrstuvwxyz" | fold -w1 | shuf -n1)
    local digit=$(echo "0123456789" | fold -w1 | shuf -n1)
    local special=$(echo "!#$%&*()_+-=[]{}|;:,.<>?" | fold -w1 | shuf -n1)

    # Combine and shuffle
    password="${base}${upper}${lower}${digit}${special}"
    password=$(echo "$password" | fold -w1 | shuf | tr -d '\n')

    # Validate password meets requirements
    if [[ ${#password} -ge 8 && ${#password} -le 64 ]] && \
       [[ "$password" =~ [A-Z] ]] && \
       [[ "$password" =~ [a-z] ]] && \
       [[ "$password" =~ [0-9] ]] && \
       [[ ! "$password" =~ [/\"\'\\@[:space:]] ]]; then
      valid=true
    fi

    ((attempts++))
  done

  if [[ "$valid" == false ]]; then
    echo "ERROR: Failed to generate valid password after $max_attempts attempts"
    exit 1
  fi

  echo "$password"
}
```

```
# Function to create secret in AWS Secrets Manager
create_secret() {
  local secret_name=$1
  local username=$2
  local password=$3
  local description=$4

  echo "Creating secret in AWS Secrets Manager: $secret_name"

  # Create the secret using AWS CLI without jq
  local secret_output=$(aws secretsmanager create-secret \
    --name "$secret_name" \
    --description "$description" \
    --secret-string "{\"username\":\"$username\",\"password\":\"$password\"}" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
 2>&1)

  if echo "$secret_output" | grep -i "error\|exception\|fail" > /dev/null; then
    echo "ERROR: Failed to create secret: $secret_output"
    return 1
  fi

  echo "Secret created successfully: $secret_name"
  return 0
}

# Function to retrieve password from AWS Secrets Manager
get_password_from_secret() {
  local secret_name=$1

  # Get the secret value and extract password using sed/grep instead of jq
  local secret_value=$(aws secretsmanager get-secret-value \
    --secret-id "$secret_name" \
    --query 'SecretString' \
    --output text 2>/dev/null)

  if [[ $? -eq 0 ]]; then
    # Extract password from JSON using sed
    echo "$secret_value" | sed -n 's/.*"password":"\([^"]*\)".*/\1/p'
  else
    echo ""
  fi
}
```

```
# Function to wait for a resource to be available
wait_for_resource() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be available..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>/dev/null)
    local status=$(echo "$output" | grep -o '"Status": "[^"]*' | cut -d'"' -f4 ||
 echo "")

    if [[ "$status" == "AVAILABLE" ]]; then
      echo "$resource_type $resource_name is now available"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name status:
 $status. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be
 available"
  return 1
}

# Function to wait for a resource to be deleted
wait_for_resource_deletion() {
  local resource_type=$1
  local resource_name=$2
  local max_attempts=$3
  local wait_seconds=$4
  local check_cmd=$5

  echo "Waiting for $resource_type $resource_name to be deleted..."

  for ((i=1; i<=$max_attempts; i++)); do
    local output=$($check_cmd 2>&1)

    if echo "$output" | grep -i "not found\|does not exist" > /dev/null; then
```

```
      echo "$resource_type $resource_name has been deleted"
      return 0
    fi

    echo "Attempt $i/$max_attempts: $resource_type $resource_name is still being
 deleted. Waiting $wait_seconds seconds..."
    sleep $wait_seconds
  done

  echo "ERROR: Timed out waiting for $resource_type $resource_name to be deleted"
  return 1
}

# Function to clean up resources
cleanup_resources() {
  echo ""
  echo "========================================="
  echo "CLEANUP CONFIRMATION"
  echo "========================================="
  echo "The following resources were created:"
  echo "- Redshift Serverless Workgroup: $WORKGROUP_NAME"
  echo "- Redshift Serverless Namespace: $NAMESPACE_NAME"
  echo "- IAM Role: $ROLE_NAME"
  echo "- Secrets Manager Secret: $SECRET_NAME"
  echo ""
  echo "Do you want to clean up all created resources? (y/n): "
  read -r CLEANUP_CHOICE

  if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the workgroup
    echo "Deleting Redshift Serverless workgroup $WORKGROUP_NAME..."
    WORKGROUP_DELETE_OUTPUT=$(aws redshift-serverless delete-workgroup --
workgroup-name "$WORKGROUP_NAME" 2>&1)
    echo "$WORKGROUP_DELETE_OUTPUT"

    # Wait for workgroup to be deleted before deleting namespace
    wait_for_resource_deletion "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-
serverless get-workgroup --workgroup-name $WORKGROUP_NAME"

    # Delete the namespace
    echo "Deleting Redshift Serverless namespace $NAMESPACE_NAME..."
```

```
    NAMESPACE_DELETE_OUTPUT=$(aws redshift-serverless delete-namespace --
namespace-name "$NAMESPACE_NAME" 2>&1)
    echo "$NAMESPACE_DELETE_OUTPUT"

    # Wait for namespace to be deleted
    wait_for_resource_deletion "namespace" "$NAMESPACE_NAME" 20 30 "aws redshift-
serverless get-namespace --namespace-name $NAMESPACE_NAME"

    # Delete the IAM role policy
    echo "Deleting IAM role policy..."
    POLICY_DELETE_OUTPUT=$(aws iam delete-role-policy --role-name "$ROLE_NAME" --
policy-name S3Access 2>&1)
    echo "$POLICY_DELETE_OUTPUT"

    # Delete the IAM role
    echo "Deleting IAM role $ROLE_NAME..."
    ROLE_DELETE_OUTPUT=$(aws iam delete-role --role-name "$ROLE_NAME" 2>&1)
    echo "$ROLE_DELETE_OUTPUT"

    # Delete the secret
    echo "Deleting Secrets Manager secret $SECRET_NAME..."
    SECRET_DELETE_OUTPUT=$(aws secretsmanager delete-secret --secret-id
 "$SECRET_NAME" --force-delete-without-recovery 2>&1)
    echo "$SECRET_DELETE_OUTPUT"

    echo "Cleanup completed."
  else
    echo "Cleanup skipped. Resources will remain in your AWS account."
  fi
}

# Check if required tools are available
if ! command -v openssl &> /dev/null; then
    echo "ERROR: openssl is required but not installed. Please install openssl to
 continue."
    exit 1
fi

# Generate unique names for resources
RANDOM_SUFFIX=$(cat /dev/urandom | tr -dc 'a-z0-9' | head -c 6)
NAMESPACE_NAME="rs-namespace-${RANDOM_SUFFIX}"
WORKGROUP_NAME="rs-workgroup-${RANDOM_SUFFIX}"
ROLE_NAME="RedshiftServerlessS3Role-${RANDOM_SUFFIX}"
SECRET_NAME="redshift-serverless-admin-${RANDOM_SUFFIX}"
```

```
DB_NAME="dev"
ADMIN_USERNAME="admin"

# Generate secure password
echo "Generating secure password..."
ADMIN_PASSWORD=$(generate_secure_password)

# Create secret in AWS Secrets Manager
create_secret "$SECRET_NAME" "$ADMIN_USERNAME" "$ADMIN_PASSWORD" "Admin
 credentials for Redshift Serverless namespace $NAMESPACE_NAME"
if [[ $? -ne 0 ]]; then
  echo "ERROR: Failed to create secret in AWS Secrets Manager"
  exit 1
fi

# Track created resources
CREATED_RESOURCES=()

echo "Using the following resource names:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- IAM Role: $ROLE_NAME"
echo "- Secret: $SECRET_NAME"
echo "- Database: $DB_NAME"
echo "- Admin Username: $ADMIN_USERNAME"
echo "- Admin Password: [STORED IN SECRETS MANAGER]"

# Step 1: Create IAM role for S3 access
echo "Creating IAM role for Redshift Serverless S3 access..."

# Create trust policy document
cat > redshift-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift-serverless.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

```
EOF

# Create S3 access policy document
cat > redshift-s3-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role $ROLE_NAME..."
ROLE_OUTPUT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://redshift-trust-policy.json 2>&1)
echo "$ROLE_OUTPUT"
check_error "$ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless
CREATED_RESOURCES+=("IAM Role: $ROLE_NAME")

# Attach S3 policy to the role
echo "Attaching S3 access policy to role $ROLE_NAME..."
POLICY_OUTPUT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 S3Access --policy-document file://redshift-s3-policy.json 2>&1)
echo "$POLICY_OUTPUT"
check_error "$POLICY_OUTPUT" "aws iam put-role-policy"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
echo "Role ARN: $ROLE_ARN"
```

```
# Step 2: Create a namespace
echo "Creating Redshift Serverless namespace $NAMESPACE_NAME..."
NAMESPACE_OUTPUT=$(aws redshift-serverless create-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --admin-username "$ADMIN_USERNAME" \
  --admin-user-password "$ADMIN_PASSWORD" \
  --db-name "$DB_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$NAMESPACE_OUTPUT"
check_error "$NAMESPACE_OUTPUT" "aws redshift-serverless create-namespace"
CREATED_RESOURCES+=("Redshift Serverless Namespace: $NAMESPACE_NAME")

# Wait for namespace to be available
wait_for_resource "namespace" "$NAMESPACE_NAME" 10 30 "aws redshift-serverless
 get-namespace --namespace-name $NAMESPACE_NAME"

# Associate IAM role with namespace
echo "Associating IAM role with namespace..."
UPDATE_NAMESPACE_OUTPUT=$(aws redshift-serverless update-namespace \
  --namespace-name "$NAMESPACE_NAME" \
  --iam-roles "$ROLE_ARN" 2>&1)
echo "$UPDATE_NAMESPACE_OUTPUT"
check_error "$UPDATE_NAMESPACE_OUTPUT" "aws redshift-serverless update-namespace"

# Step 3: Create a workgroup
echo "Creating Redshift Serverless workgroup $WORKGROUP_NAME..."
WORKGROUP_OUTPUT=$(aws redshift-serverless create-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --namespace-name "$NAMESPACE_NAME" \
  --base-capacity 8 \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-serverless 2>&1)
echo "$WORKGROUP_OUTPUT"
check_error "$WORKGROUP_OUTPUT" "aws redshift-serverless create-workgroup"
CREATED_RESOURCES+=("Redshift Serverless Workgroup: $WORKGROUP_NAME")

# Wait for workgroup to be available
wait_for_resource "workgroup" "$WORKGROUP_NAME" 20 30 "aws redshift-serverless
 get-workgroup --workgroup-name $WORKGROUP_NAME"

# Get workgroup endpoint
WORKGROUP_ENDPOINT=$(aws redshift-serverless get-workgroup \
  --workgroup-name "$WORKGROUP_NAME" \
  --query 'workgroup.endpoint.address' \
  --output text)
```

```
echo "Workgroup endpoint: $WORKGROUP_ENDPOINT"

# Wait additional time for the endpoint to be fully operational
echo "Waiting for endpoint to be fully operational..."
sleep 60

# Step 4: Create tables for sample data
echo "Creating tables for sample data..."

# Create users table
echo "Creating users table..."
USERS_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE users(
    userid INTEGER NOT NULL DISTKEY SORTKEY,
    username CHAR(8),
    firstname VARCHAR(30),
    lastname VARCHAR(30),
    city VARCHAR(30),
    state CHAR(2),
    email VARCHAR(100),
    phone CHAR(14),
    likesports BOOLEAN,
    liketheatre BOOLEAN,
    likeconcerts BOOLEAN,
    likejazz BOOLEAN,
    likeclassical BOOLEAN,
    likeopera BOOLEAN,
    likerock BOOLEAN,
    likevegas BOOLEAN,
    likebroadway BOOLEAN,
    likemusicals BOOLEAN
  );" 2>&1)
echo "$USERS_TABLE_OUTPUT"
check_error "$USERS_TABLE_OUTPUT" "aws redshift-data execute-statement (users
 table)"
USERS_QUERY_ID=$(echo "$USERS_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for users table creation to complete..."
sleep 5
```

```
# Create event table
echo "Creating event table..."
EVENT_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE event(
    eventid INTEGER NOT NULL DISTKEY,
    venueid SMALLINT NOT NULL,
    catid SMALLINT NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    eventname VARCHAR(200),
    starttime TIMESTAMP
  );" 2>&1)
echo "$EVENT_TABLE_OUTPUT"
check_error "$EVENT_TABLE_OUTPUT" "aws redshift-data execute-statement (event
 table)"
EVENT_QUERY_ID=$(echo "$EVENT_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for query to complete
echo "Waiting for event table creation to complete..."
sleep 5

# Create sales table
echo "Creating sales table..."
SALES_TABLE_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "CREATE TABLE sales(
    salesid INTEGER NOT NULL,
    listid INTEGER NOT NULL DISTKEY,
    sellerid INTEGER NOT NULL,
    buyerid INTEGER NOT NULL,
    eventid INTEGER NOT NULL,
    dateid SMALLINT NOT NULL SORTKEY,
    qtysold SMALLINT NOT NULL,
    pricepaid DECIMAL(8,2),
    commission DECIMAL(8,2),
    saletime TIMESTAMP
  );" 2>&1)
echo "$SALES_TABLE_OUTPUT"
check_error "$SALES_TABLE_OUTPUT" "aws redshift-data execute-statement (sales
 table)"
```

```
SALES_QUERY_ID=$(echo "$SALES_TABLE_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"'
 -f4)

# Wait for tables to be created
echo "Waiting for tables to be created..."
sleep 10

# Step 5: Load sample data from Amazon S3
echo "Loading sample data from Amazon S3..."

# Load data into users table
echo "Loading data into users table..."
USERS_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY users
    FROM 's3://redshift-downloads/tickit/allusers_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$USERS_LOAD_OUTPUT"
check_error "$USERS_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 users)"
USERS_LOAD_QUERY_ID=$(echo "$USERS_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for users data loading to complete..."
sleep 10

# Load data into event table
echo "Loading data into event table..."
EVENT_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY event
    FROM 's3://redshift-downloads/tickit/allevents_pipe.txt'
    DELIMITER '|'
    TIMEFORMAT 'YYYY-MM-DD HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$EVENT_LOAD_OUTPUT"
```

```
check_error "$EVENT_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 event)"
EVENT_LOAD_QUERY_ID=$(echo "$EVENT_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for event data loading to complete..."
sleep 10

# Load data into sales table
echo "Loading data into sales table..."
SALES_LOAD_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "COPY sales
    FROM 's3://redshift-downloads/tickit/sales_tab.txt'
    DELIMITER '\t'
    TIMEFORMAT 'MM/DD/YYYY HH:MI:SS'
    IGNOREHEADER 1
    IAM_ROLE '$ROLE_ARN';" 2>&1)
echo "$SALES_LOAD_OUTPUT"
check_error "$SALES_LOAD_OUTPUT" "aws redshift-data execute-statement (load
 sales)"
SALES_LOAD_QUERY_ID=$(echo "$SALES_LOAD_OUTPUT" | grep -o '"Id": "[^"]*' | cut -
d'"' -f4)

# Wait for data loading to complete
echo "Waiting for sales data loading to complete..."
sleep 30

# Step 6: Run sample queries
echo "Running sample queries..."

# Query 1: Find top 10 buyers by quantity
echo "Running query: Find top 10 buyers by quantity..."
QUERY1_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT firstname, lastname, total_quantity
    FROM (SELECT buyerid, sum(qtysold) total_quantity
          FROM sales
          GROUP BY buyerid
          ORDER BY total_quantity desc limit 10) Q, users
    WHERE Q.buyerid = userid
```

```
    ORDER BY Q.total_quantity desc;" 2>&1)
echo "$QUERY1_OUTPUT"
check_error "$QUERY1_OUTPUT" "aws redshift-data execute-statement (query 1)"
QUERY1_ID=$(echo "$QUERY1_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 1 to complete..."
sleep 10

# Get query 1 results
echo "Getting results for query 1..."
QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
echo "$QUERY1_STATUS_OUTPUT"
check_error "$QUERY1_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 1)"

QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY1_STATUS" == "FINISHED" ]; then
  QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID" 2>&1)
  echo "Query 1 Results:"
  echo "$QUERY1_RESULTS"
else
  echo "Query 1 is not yet complete. Status: $QUERY1_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
  QUERY1_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY1_ID"
 2>&1)
  QUERY1_STATUS=$(echo "$QUERY1_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY1_STATUS" == "FINISHED" ]; then
    QUERY1_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY1_ID"
 2>&1)
    echo "Query 1 Results:"
    echo "$QUERY1_RESULTS"
  else
    echo "Query 1 is still not complete. Status: $QUERY1_STATUS"
  fi
fi
```

```
# Query 2: Find events in the 99.9 percentile in terms of all time total sales
echo "Running query: Find events in the 99.9 percentile in terms of all time
 total sales..."
QUERY2_OUTPUT=$(aws redshift-data execute-statement \
  --database "$DB_NAME" \
  --workgroup-name "$WORKGROUP_NAME" \
  --sql "SELECT eventname, total_price
    FROM (SELECT eventid, total_price, ntile(1000) over(order by total_price
 desc) as percentile
          FROM (SELECT eventid, sum(pricepaid) total_price
                FROM sales
                GROUP BY eventid)) Q, event E
    WHERE Q.eventid = E.eventid
    AND percentile = 1
    ORDER BY total_price desc;" 2>&1)
echo "$QUERY2_OUTPUT"
check_error "$QUERY2_OUTPUT" "aws redshift-data execute-statement (query 2)"
QUERY2_ID=$(echo "$QUERY2_OUTPUT" | grep -o '"Id": "[^"]*' | cut -d'"' -f4)

# Wait for query to complete
echo "Waiting for query 2 to complete..."
sleep 10

# Get query 2 results
echo "Getting results for query 2..."
QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
echo "$QUERY2_STATUS_OUTPUT"
check_error "$QUERY2_STATUS_OUTPUT" "aws redshift-data describe-statement (query
 2)"

QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut -
d'"' -f4)
if [ "$QUERY2_STATUS" == "FINISHED" ]; then
  QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID" 2>&1)
  echo "Query 2 Results:"
  echo "$QUERY2_RESULTS"
else
  echo "Query 2 is not yet complete. Status: $QUERY2_STATUS"
  echo "Waiting additional time for query to complete..."
  sleep 20

  # Check again
```

```
  QUERY2_STATUS_OUTPUT=$(aws redshift-data describe-statement --id "$QUERY2_ID"
 2>&1)
  QUERY2_STATUS=$(echo "$QUERY2_STATUS_OUTPUT" | grep -o '"Status": "[^"]*' | cut
 -d'"' -f4)

  if [ "$QUERY2_STATUS" == "FINISHED" ]; then
    QUERY2_RESULTS=$(aws redshift-data get-statement-result --id "$QUERY2_ID"
 2>&1)
    echo "Query 2 Results:"
    echo "$QUERY2_RESULTS"
  else
    echo "Query 2 is still not complete. Status: $QUERY2_STATUS"
  fi
fi

# Summary
echo ""
echo "=========================================="
echo "TUTORIAL SUMMARY"
echo "=========================================="
echo "You have successfully:"
echo "1. Created a Redshift Serverless namespace and workgroup"
echo "2. Created an IAM role with S3 access permissions"
echo "3. Stored admin credentials securely in AWS Secrets Manager"
echo "4. Created tables for sample data"
echo "5. Loaded sample data from Amazon S3"
echo "6. Run sample queries on the data"
echo ""
echo "Redshift Serverless Resources:"
echo "- Namespace: $NAMESPACE_NAME"
echo "- Workgroup: $WORKGROUP_NAME"
echo "- Database: $DB_NAME"
echo "- Endpoint: $WORKGROUP_ENDPOINT"
echo "- Credentials Secret: $SECRET_NAME"
echo ""
echo "To connect to your Redshift Serverless database using SQL tools:"
echo "- Host: $WORKGROUP_ENDPOINT"
echo "- Database: $DB_NAME"
echo "- Username: $ADMIN_USERNAME"
echo "- Password: Retrieve from AWS Secrets Manager secret '$SECRET_NAME'"
echo ""
echo "To retrieve the password from Secrets Manager (without jq):"
echo "aws secretsmanager get-secret-value --secret-id $SECRET_NAME --query
 'SecretString' --output text | sed -n 's/.*\"password\":\"\([^\"]*\)\".*/\1/p'"
```

```
echo ""

# Clean up temporary files
rm -f redshift-trust-policy.json redshift-s3-policy.json

# Clean up resources
cleanup_resources

echo "Tutorial completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - CreateNamespace
  - CreateRole
  - CreateSecret
  - CreateWorkgroup
  - DeleteNamespace
  - DeleteRole
  - DeleteRolePolicy
  - DeleteSecret
  - DeleteWorkgroup
  - GetNamespace
  - GetRole
  - GetSecretValue
  - GetWorkgroup
  - PutRolePolicy
  - UpdateNamespace

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

IoT Device Defender 入门

以下代码示例展示了如何：

- 创建所需的 IAM 角色

- 启用 IoT Device Defender 审计检查
- 运行按需审核
- 创建缓解操作
- 将缓解操作应用于调查发现
- 设置 SNS 通知（可选）
- 启用 IoT 日志记录

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS IoT Device Defender Getting Started Script
# This script demonstrates how to use AWS IoT Device Defender to enable audit
 checks,
# view audit results, create mitigation actions, and apply them to findings.

set -euo pipefail

# Set up logging
LOG_FILE="iot-device-defender-script-$(date +%Y%m%d%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "================================================="
echo "AWS IoT Device Defender Getting Started Script"
echo "================================================="
echo "Starting script execution at $(date)"
echo ""

# Function to check for errors in command output
check_error() {
```

```
    if echo "$1" | grep -iE "An error occurred|Exception|Failed|usage: aws" > /
dev/null; then
        echo "ERROR: Command failed with the following output:"
        echo "$1"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values using jq
extract_json_value() {
    local json="$1"
    local key="$2"
    echo "$json" | jq -r ".${key} // empty" 2>/dev/null || echo ""
}

# Function to validate JSON
validate_json() {
    local json="$1"
    echo "$json" | jq empty 2>/dev/null
}

# Function to check AWS CLI availability
check_aws_cli() {
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed or not in PATH"
        return 1
    fi
    if ! command -v jq &> /dev/null; then
        echo "ERROR: jq is not installed or not in PATH"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    local account_id
    account_id=$(aws sts get-caller-identity --query 'Account' --output text 2>/
dev/null) || true
    if [ -z "$account_id" ]; then
        echo "ERROR: Could not retrieve AWS account ID"
        return 1
    fi
```

```
    echo "$account_id"
    return 0
}

# Function to create IAM roles with retry logic
create_iam_role() {
    local ROLE_NAME=$1
    local TRUST_POLICY=$2
    local MANAGED_POLICY=$3
    local RETRY_COUNT=0
    local MAX_RETRIES=3

    echo "Creating IAM role: $ROLE_NAME"

    # Validate trust policy JSON
    if ! validate_json "$TRUST_POLICY"; then
        echo "ERROR: Invalid trust policy JSON for role $ROLE_NAME"
        return 1
    fi

    # Check if role already exists
    if aws iam get-role --role-name "$ROLE_NAME" >/dev/null 2>&1; then
        echo "Role $ROLE_NAME already exists, skipping creation"
        ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn'
 --output text 2>/dev/null) || true
        if [ -z "$ROLE_ARN" ]; then
            echo "ERROR: Could not retrieve ARN for existing role $ROLE_NAME"
            return 1
        fi
        echo "Role ARN: $ROLE_ARN"
        return 0
    fi

    # Create the role with trust policy and retry logic
    while [ $RETRY_COUNT -lt $MAX_RETRIES ]; do
        ROLE_RESULT=$(aws iam create-role \
            --role-name "$ROLE_NAME" \
            --assume-role-policy-document "$TRUST_POLICY" 2>&1) || true

        if check_error "$ROLE_RESULT"; then
            break
        fi

        RETRY_COUNT=$((RETRY_COUNT + 1))
```

```
        if [ $RETRY_COUNT -lt $MAX_RETRIES ]; then
            echo "Retrying role creation (attempt $((RETRY_COUNT + 1))/
$MAX_RETRIES)..."
            sleep $((RETRY_COUNT * 2))
        fi
    done

    if ! check_error "$ROLE_RESULT"; then
        echo "Failed to create role $ROLE_NAME after $MAX_RETRIES attempts"
        return 1
    fi

    aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

    # For IoT logging role, create an inline policy instead of using a managed
 policy
    if [[ "$ROLE_NAME" == "AWSIoTLoggingRole" ]]; then
        local LOGGING_POLICY
        LOGGING_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:PutLogEvents",
                "logs:PutMetricFilter",
                "logs:PutRetentionPolicy",
                "logs:GetLogEvents",
                "logs:DescribeLogStreams"
            ],
            "Resource": "arn:aws:logs:*:*:*"
        }
    ]
}
EOF
)

        if ! validate_json "$LOGGING_POLICY"; then
            echo "ERROR: Invalid logging policy JSON"
            return 1
```

```
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$LOGGING_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    elif [[ "$ROLE_NAME" == "IoTMitigationActionErrorLoggingRole" ]]; then
        local MITIGATION_POLICY
        MITIGATION_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iot:UpdateCACertificate",
                "iot:UpdateCertificate",
                "iot:SetV2LoggingOptions",
                "iot:SetLoggingOptions",
                "iot:AddThingToThingGroup"
            ],
            "Resource": "arn:aws:iot:*:*:*"
        },
        {
            "Effect": "Allow",
            "Action": "iam:PassRole",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:PassedToService": "iot.amazonaws.com"
                }
            }
        }
    ]
}
EOF
)

        if ! validate_json "$MITIGATION_POLICY"; then
```

```
            echo "ERROR: Invalid mitigation policy JSON"
            return 1
        fi

        POLICY_RESULT=$(aws iam put-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-name "${ROLE_NAME}Policy" \
            --policy-document "$MITIGATION_POLICY" 2>&1) || true

        if ! check_error "$POLICY_RESULT"; then
            echo "Failed to attach inline policy to role $ROLE_NAME"
            return 1
        fi
    else
        # Attach managed policy to role if provided
        if [ -n "$MANAGED_POLICY" ]; then
            ATTACH_RESULT=$(aws iam attach-role-policy \
                --role-name "$ROLE_NAME" \
                --policy-arn "$MANAGED_POLICY" 2>&1) || true

            if ! check_error "$ATTACH_RESULT"; then
                echo "Failed to attach policy to role $ROLE_NAME"
                return 1
            fi
        fi
    fi

    echo "Role $ROLE_NAME created successfully"

    # Get the role ARN with error handling
    ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --
output text 2>/dev/null) || true
    if [ -z "$ROLE_ARN" ]; then
        echo "ERROR: Could not retrieve ARN for newly created role $ROLE_NAME"
        return 1
    fi
    echo "Role ARN: $ROLE_ARN"
    return 0
}

# Array to store created resources for cleanup
declare -a CREATED_RESOURCES

# Validate prerequisites
```

```
echo "Validating prerequisites..."
if ! check_aws_cli; then
    echo "ERROR: Prerequisites not met"
    exit 1
fi

ACCOUNT_ID=$(get_account_id) || exit 1
echo "AWS Account ID: $ACCOUNT_ID"
echo ""

# Step 1: Create IAM roles needed for the tutorial
echo "=================================================="
echo "Step 1: Creating required IAM roles"
echo "=================================================="

# Create IoT Device Defender Audit role
IOT_DEFENDER_AUDIT_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTDeviceDefenderAuditRole"
 "$IOT_DEFENDER_AUDIT_TRUST_POLICY" "arn:aws:iam::aws:policy/service-role/
AWSIoTDeviceDefenderAudit"; then
    echo "ERROR: Failed to create audit role"
    exit 1
fi
AUDIT_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTDeviceDefenderAuditRole")

# Create IoT Logging role
IOT_LOGGING_TRUST_POLICY=$(cat <<'EOF'
{
```

```
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "AWSIoTLoggingRole" "$IOT_LOGGING_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create logging role"
    exit 1
fi
LOGGING_ROLE_ARN=$ROLE_ARN
CREATED_RESOURCES+=("IAM Role: AWSIoTLoggingRole")

# Create IoT Mitigation Action role
IOT_MITIGATION_TRUST_POLICY=$(cat <<'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "iot.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
)

if ! create_iam_role "IoTMitigationActionErrorLoggingRole"
 "$IOT_MITIGATION_TRUST_POLICY" ""; then
    echo "ERROR: Failed to create mitigation role"
    exit 1
fi
MITIGATION_ROLE_ARN=$ROLE_ARN
```

```
CREATED_RESOURCES+=("IAM Role: IoTMitigationActionErrorLoggingRole")

# Wait for IAM role propagation
echo "Waiting for IAM role propagation..."
sleep 5

# Step 2: Enable audit checks
echo ""
echo "================================================="
echo "Step 2: Enabling AWS IoT Device Defender audit checks"
echo "================================================="

# Get current audit configuration
echo "Getting current audit configuration..."
CURRENT_CONFIG=$(aws iot describe-account-audit-configuration --output json 2>&1)
 || true
if validate_json "$CURRENT_CONFIG"; then
    echo "$CURRENT_CONFIG" | jq '.' 2>/dev/null || echo "Could not parse current
 configuration"
fi

# Enable specific audit checks with proper JSON escaping
echo "Enabling audit checks..."
AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":true}}'

if ! validate_json "$AUDIT_CONFIG"; then
    echo "ERROR: Invalid audit configuration JSON"
    exit 1
fi

UPDATE_RESULT=$(aws iot update-account-audit-configuration \
  --role-arn "$AUDIT_ROLE_ARN" \
  --audit-check-configurations "$AUDIT_CONFIG" 2>&1) || true

if ! check_error "$UPDATE_RESULT"; then
    echo "Failed to update audit configuration"
    exit 1
fi

echo "Audit checks enabled successfully"

# Step 3: Run an on-demand audit
echo ""
echo "================================================="
```

```
echo "Step 3: Running an on-demand audit"
echo "================================================="

echo "Starting on-demand audit task..."
AUDIT_TASK_RESULT=$(aws iot start-on-demand-audit-task \
  --target-check-names LOGGING_DISABLED_CHECK --output json 2>&1) || true

if ! check_error "$AUDIT_TASK_RESULT"; then
    echo "Failed to start on-demand audit task"
    exit 1
fi

TASK_ID=$(extract_json_value "$AUDIT_TASK_RESULT" "taskId")
if [ -z "$TASK_ID" ]; then
    echo "ERROR: Could not extract task ID from response"
    exit 1
fi

echo "Audit task started with ID: $TASK_ID"
CREATED_RESOURCES+=("Audit Task: $TASK_ID")

# Tag the audit task via IoT service
aws iot tag-resource --resource-arn "arn:aws:iot:$(aws configure get region):
${ACCOUNT_ID}:audittask/${TASK_ID}" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-device-defender-gs 2>&1 || true

# Wait for the audit task to complete
echo "Waiting for audit task to complete (this may take a few minutes)..."
TASK_STATUS="IN_PROGRESS"
TIMEOUT=0
MAX_TIMEOUT=600
POLL_INTERVAL=15

while [ "$TASK_STATUS" != "COMPLETED" ]; do
    if [ $TIMEOUT -ge $MAX_TIMEOUT ]; then
        echo "WARNING: Audit task did not complete within ${MAX_TIMEOUT} seconds,
 continuing..."
        break
    fi

    sleep "$POLL_INTERVAL"
    TIMEOUT=$((TIMEOUT + POLL_INTERVAL))
```

```
    TASK_DETAILS=$(aws iot describe-audit-task --task-id "$TASK_ID" --output json
 2>&1) || true
    if validate_json "$TASK_DETAILS"; then
        TASK_STATUS=$(extract_json_value "$TASK_DETAILS" "taskStatus")
        echo "Current task status: $TASK_STATUS (elapsed: ${TIMEOUT}s)"

        if [ "$TASK_STATUS" = "FAILED" ]; then
            echo "WARNING: Audit task failed, continuing with script..."
            FAILURE_REASON=$(extract_json_value "$TASK_DETAILS"
 "taskStatistics.failedChecksNotApplicable")
            if [ -n "$FAILURE_REASON" ]; then
                echo "Reason: $FAILURE_REASON"
            fi
            break
        fi
    else
        echo "WARNING: Could not parse task details, retrying..."
    fi
done

echo "Audit task processing completed"

# Get audit findings (non-blocking)
echo "Getting audit findings..."
FINDINGS=$(aws iot list-audit-findings \
  --task-id "$TASK_ID" --output json 2>&1) || true

if validate_json "$FINDINGS"; then
    FINDING_COUNT=$(echo "$FINDINGS" | jq '.findings | length' 2>/dev/null ||
 echo "0")
    echo "Audit findings count: $FINDING_COUNT"
    if [ "$FINDING_COUNT" -gt 0 ]; then
        echo "Sample finding:"
        echo "$FINDINGS" | jq '.findings[0]' 2>/dev/null || echo "Could not parse
 finding"
    fi
else
    echo "WARNING: Could not parse audit findings response"
    FINDINGS='{"findings":[]}'
fi

# Check if we have any non-compliant findings
FINDING_ID=$(extract_json_value "$FINDINGS" "findings[0].findingId")
if [ -n "$FINDING_ID" ]; then
```

```
    echo "Found non-compliant finding with ID: $FINDING_ID"
    HAS_FINDINGS=true
else
    echo "No non-compliant findings detected"
    HAS_FINDINGS=false
fi

# Step 4: Create a mitigation action
echo ""
echo "=================================================="
echo "Step 4: Creating a mitigation action"
echo "=================================================="

# Check if mitigation action already exists and delete it
if aws iot describe-mitigation-action --action-name "EnableErrorLoggingAction" >/
dev/null 2>&1; then
    echo "Mitigation action 'EnableErrorLoggingAction' already exists, deleting
 it first..."
    aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction"
 2>&1 || true
    sleep 2
fi

echo "Creating mitigation action to enable AWS IoT logging..."

# Build mitigation action parameters JSON
MITIGATION_PARAMS=$(cat <<EOF
{
  "enableIoTLoggingParams": {
    "roleArnForLogging": "$LOGGING_ROLE_ARN",
    "logLevel": "ERROR"
  }
}
EOF
)

if ! validate_json "$MITIGATION_PARAMS"; then
    echo "ERROR: Invalid mitigation parameters JSON"
    exit 1
fi

MITIGATION_RESULT=$(aws iot create-mitigation-action \
  --action-name "EnableErrorLoggingAction" \
  --role-arn "$MITIGATION_ROLE_ARN" \
```

```
  --action-params "$MITIGATION_PARAMS" --output json 2>&1) || true

if ! check_error "$MITIGATION_RESULT"; then
    echo "Failed to create mitigation action"
    exit 1
fi

if validate_json "$MITIGATION_RESULT"; then
    echo "Mitigation action created successfully"
    MITIGATION_ACTION_ARN=$(extract_json_value "$MITIGATION_RESULT" "actionArn")
    if [ -n "$MITIGATION_ACTION_ARN" ]; then
        echo "Mitigation Action ARN: $MITIGATION_ACTION_ARN"
        aws iot tag-resource --resource-arn "$MITIGATION_ACTION_ARN" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs 2>&1
 || true
    fi
else
    echo "WARNING: Could not validate mitigation action response, but action may
 have been created"
fi

CREATED_RESOURCES+=("Mitigation Action: EnableErrorLoggingAction")

# Step 5: Apply mitigation action to findings (if any)
if [ "$HAS_FINDINGS" = true ]; then
    echo ""
    echo "=================================================="
    echo "Step 5: Applying mitigation action to findings"
    echo "=================================================="

    MITIGATION_TASK_ID="MitigationTask-$(date +%s)"
    echo "Starting mitigation actions task with ID: $MITIGATION_TASK_ID"

    # Build target JSON
    TARGET_JSON=$(cat <<EOF
{
  "findingIds": ["$FINDING_ID"]
}
EOF
)

    if ! validate_json "$TARGET_JSON"; then
        echo "ERROR: Invalid target JSON"
        exit 1
```

```
    fi

    # Build audit check to actions mapping JSON
    AUDIT_CHECK_MAPPING=$(cat <<EOF
{
  "LOGGING_DISABLED_CHECK": ["EnableErrorLoggingAction"]
}
EOF
)

    if ! validate_json "$AUDIT_CHECK_MAPPING"; then
        echo "ERROR: Invalid audit check mapping JSON"
        exit 1
    fi

    MITIGATION_TASK_RESULT=$(aws iot start-audit-mitigation-actions-task \
      --task-id "$MITIGATION_TASK_ID" \
      --target "$TARGET_JSON" \
      --audit-check-to-actions-mapping "$AUDIT_CHECK_MAPPING" --output json 2>&1)
 || true

    if ! check_error "$MITIGATION_TASK_RESULT"; then
        echo "WARNING: Failed to start mitigation actions task, continuing..."
    else
        echo "Mitigation actions task started successfully"
        CREATED_RESOURCES+=("Mitigation Task: $MITIGATION_TASK_ID")
    fi
else
    echo ""
    echo "=================================================="
    echo "Step 5: Skipping mitigation action application (no findings)"
    echo "=================================================="
fi

# Step 6: Set up SNS notifications (optional)
echo ""
echo "=================================================="
echo "Step 6: Setting up SNS notifications"
echo "=================================================="

# Check if SNS topic already exists
SNS_TOPICS=$(aws sns list-topics --output json 2>&1) || true
TOPIC_ARN=""
if validate_json "$SNS_TOPICS"; then
```

```
    TOPIC_ARN=$(echo "$SNS_TOPICS" | jq -r '.Topics[] | select(.TopicArn |
 contains("IoTDDNotifications")) | .TopicArn' 2>/dev/null | head -1 || echo "")
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "SNS topic 'IoTDDNotifications' already exists, using existing topic..."
    echo "Topic ARN: $TOPIC_ARN"
else
    echo "Creating SNS topic for notifications..."
    SNS_RESULT=$(aws sns create-topic --name "IoTDDNotifications" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-iot-device-defender-gs --
output json 2>&1) || true

    if ! check_error "$SNS_RESULT"; then
        echo "WARNING: Failed to create SNS topic, continuing..."
        SNS_RESULT=""
    else
        TOPIC_ARN=$(extract_json_value "$SNS_RESULT" "TopicArn")
        if [ -n "$TOPIC_ARN" ]; then
            echo "SNS topic created with ARN: $TOPIC_ARN"
            CREATED_RESOURCES+=("SNS Topic: IoTDDNotifications")
        fi
    fi
fi

if [ -n "$TOPIC_ARN" ]; then
    echo "Updating audit configuration to enable SNS notifications..."

    # Build SNS notification configuration JSON
    SNS_CONFIG=$(cat <<EOF
{
  "SNS": {
    "targetArn": "$TOPIC_ARN",
    "roleArn": "$AUDIT_ROLE_ARN",
    "enabled": true
  }
}
EOF
)

    if ! validate_json "$SNS_CONFIG"; then
        echo "ERROR: Invalid SNS configuration JSON"
        exit 1
    fi
```

```
    SNS_UPDATE_RESULT=$(aws iot update-account-audit-configuration \
      --audit-notification-target-configurations "$SNS_CONFIG" 2>&1) || true

    if ! check_error "$SNS_UPDATE_RESULT"; then
        echo "WARNING: Failed to update audit configuration for SNS
 notifications"
    else
        echo "SNS notifications enabled successfully"
    fi
else
    echo "Skipping SNS configuration due to topic creation failure"
fi

# Step 7: Enable AWS IoT logging
echo ""
echo "=================================================="
echo "Step 7: Enabling AWS IoT logging"
echo "=================================================="

echo "Setting up AWS IoT logging options..."

LOGGING_RESULT=$(aws iot set-v2-logging-options \
  --role-arn "$LOGGING_ROLE_ARN" \
  --default-log-level "ERROR" 2>&1) || true

if ! check_error "$LOGGING_RESULT"; then
    echo "V2 logging setup failed, trying v1 logging..."

    V1_LOGGING_CONFIG=$(cat <<EOF
{
  "roleArn": "$LOGGING_ROLE_ARN",
  "logLevel": "ERROR"
}
EOF
)

    if ! validate_json "$V1_LOGGING_CONFIG"; then
        echo "ERROR: Invalid v1 logging configuration JSON"
        exit 1
    fi

    LOGGING_RESULT_V1=$(aws iot set-logging-options \
      --logging-options-payload "$V1_LOGGING_CONFIG" 2>&1) || true
```

```
    if ! check_error "$LOGGING_RESULT_V1"; then
        echo "WARNING: Failed to set up AWS IoT logging with both v1 and v2
 methods, continuing..."
    else
        echo "AWS IoT v1 logging enabled successfully"
    fi
else
    echo "AWS IoT v2 logging enabled successfully"
fi

# Verify logging is enabled
echo "Verifying logging configuration..."
LOGGING_CONFIG=$(aws iot get-v2-logging-options --output json 2>&1) || true
if [ -n "$LOGGING_CONFIG" ] && ! check_error "$LOGGING_CONFIG" && validate_json
 "$LOGGING_CONFIG"; then
    echo "Logging configuration verified"
    echo "$LOGGING_CONFIG" | jq '.' 2>/dev/null || echo "Configuration retrieved
 but could not display details"
else
    echo "Could not verify logging configuration, but setup completed"
fi

# Script completed successfully
echo ""
echo "================================================="
echo "AWS IoT Device Defender setup completed successfully!"
echo "================================================="
echo "The following resources were created:"
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "- $resource"
done
echo ""

# Cleanup phase
echo "=========================================="
echo "CLEANUP"
echo "=========================================="
echo "Starting automatic cleanup of resources..."
echo "Waiting 10 seconds before cleanup to allow resource stabilization..."
sleep 10

# Disable AWS IoT logging
echo "Disabling AWS IoT logging..."
```

```
DISABLE_V2_RESULT=$(aws iot set-v2-logging-options \
  --default-log-level "DISABLED" 2>&1) || true

if check_error "$DISABLE_V2_RESULT"; then
    echo "V2 logging disabled successfully"
else
    echo "Attempting v1 logging disable..."

    V1_DISABLE_CONFIG=$(cat <<'EOF'
{
  "logLevel": "DISABLED"
}
EOF
)

    DISABLE_V1_RESULT=$(aws iot set-logging-options \
      --logging-options-payload "$V1_DISABLE_CONFIG" 2>&1) || true

    if check_error "$DISABLE_V1_RESULT"; then
        echo "V1 logging disabled successfully"
    else
        echo "WARNING: Could not disable logging"
    fi
fi

# Delete mitigation action
echo "Deleting mitigation action..."
aws iot delete-mitigation-action --action-name "EnableErrorLoggingAction" 2>&1 ||
 true

# Reset audit configuration
echo "Resetting IoT Device Defender audit configuration..."
RESET_AUDIT_CONFIG='{"LOGGING_DISABLED_CHECK":{"enabled":false}}'
aws iot update-account-audit-configuration \
  --audit-check-configurations "$RESET_AUDIT_CONFIG" 2>&1 || true

# Delete SNS topic
echo "Deleting SNS topic..."
if [ -n "${TOPIC_ARN:-}" ] && [ "$TOPIC_ARN" != "null" ]; then
    aws sns delete-topic --topic-arn "$TOPIC_ARN" 2>&1 || true
fi

# Clean up IAM roles with improved error handling
```

```
echo "Cleaning up IAM roles..."

cleanup_role() {
    local role_name=$1
    echo "Cleaning up role: $role_name"

    if aws iam get-role --role-name "$role_name" >/dev/null 2>&1; then
        ROLE_POLICIES=$(aws iam list-role-policies --role-name "$role_name" --
output json 2>&1 || echo '{"PolicyNames":[]}')
        if validate_json "$ROLE_POLICIES"; then
            while IFS= read -r policy_name; do
                if [ -n "$policy_name" ] && [ "$policy_name" != "null" ]; then
                    echo "  Deleting inline policy: $policy_name"
                    aws iam delete-role-policy \
                        --role-name "$role_name" \
                        --policy-name "$policy_name" 2>&1 || true
                fi
            done < <(echo "$ROLE_POLICIES" | jq -r '.PolicyNames[]' 2>/dev/null
 || echo "")
        fi

        ATTACHED_POLICIES=$(aws iam list-attached-role-policies --role-name
 "$role_name" --output json 2>&1 || echo '{"AttachedPolicies":[]}')
        if validate_json "$ATTACHED_POLICIES"; then
            while IFS= read -r policy_arn; do
                if [ -n "$policy_arn" ] && [ "$policy_arn" != "null" ]; then
                    echo "  Detaching managed policy: $policy_arn"
                    aws iam detach-role-policy \
                        --role-name "$role_name" \
                        --policy-arn "$policy_arn" 2>&1 || true
                fi
            done < <(echo "$ATTACHED_POLICIES" | jq -r
 '.AttachedPolicies[].PolicyArn' 2>/dev/null || echo "")
        fi

        echo "  Deleting role: $role_name"
        aws iam delete-role --role-name "$role_name" 2>&1 || true
    else
        echo "  Role $role_name does not exist or already deleted"
    fi
}

cleanup_role "AWSIoTDeviceDefenderAuditRole"
cleanup_role "AWSIoTLoggingRole"
```

```
cleanup_role "IoTMitigationActionErrorLoggingRole"

echo "Cleanup completed successfully"

echo ""
echo "Script execution completed at $(date)"
echo "Log file: $LOG_FILE"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateMitigationAction
  - CreateRole
  - CreateTopic
  - DeleteAccountAuditConfiguration
  - DeleteMitigationAction
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeAccountAuditConfiguration
  - DescribeAuditTask
  - DetachRolePolicy
  - GetLoggingOptions
  - GetRole
  - GetV2LoggingOptions
  - ListAuditFindings
  - ListAuditMitigationActionsTasks
  - ListMitigationActions
  - ListRolePolicies
  - ListTopics
  - PutRolePolicy
  - SetLoggingOptions
  - SetV2LoggingOptions
  - StartAuditMitigationActionsTask

- StartOnDemandAuditTask
- UpdateAccountAuditConfiguration

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用 Amazon EKS

以下代码示例展示了如何：

- 为 EKS 集群创建 VPC
- 为 EKS 集群创建 IAM 角色
- 创建您的 EKS 集群
- 配置 kubectl 以与集群通信
- 创建托管节点组
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
```

```
echo "All commands and outputs will be logged to $LOG_FILE"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
```

```
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
```

```
        echo "Deleting node group: $NODEGROUP_NAME"
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
```

```
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."
```

```
# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
```

```
fi
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
```

```
fi
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
```

```
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."
```

```
echo "Cluster information:"
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="

# Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE
```

```
if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole

- ListNodegroups
- UpdateKubeconfig
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon MSK 入门

以下代码示例展示了如何：

- 创建 MSK 集群
- 为 MSK 访问创建 IAM 权限
- 创建客户端计算机
- 获取引导代理
- 设置客户端计算机
- 创建主题并生成/使用数据
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
```

```
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
```

```
            ;;
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)
```

```
                if [ -n "$sg_details" ]; then
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
```

```
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
 removed
```

```
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
```

```
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
```

```
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
```

```
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"
```

```
# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
```

```
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
```

```
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
```

```
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"
```

```
# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi
```

```
echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
```

```
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)
```

```
if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))
```

```
    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
```

```
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
```

```
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF
```

```
# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "============================================="

# Display summary of created resources
echo ""
echo "============================================="
echo "RESOURCE SUMMARY"
echo "============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
```

```
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。

- AddRoleToInstanceProfile
- AttachRolePolicy
- AuthorizeSecurityGroupIngress
- CreateCluster
- CreateInstanceProfile
- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole

- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon Redshift 预置集群入门

以下代码示例展示了如何：

- 创建 Redshift 集群
- 为 S3 访问创建 IAM 角色
- 创建表并加载数据
- 运行示例查询
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Redshift Provisioned Cluster Tutorial Script
# This script creates a Redshift cluster, loads sample data, runs queries, and
 cleans up resources
# Version 4: Security improvements and best practices
```

```
set -euo pipefail

# Set up logging
LOG_FILE="redshift_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Redshift tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to handle errors
handle_error() {
    echo "ERROR: $1" >&2
    echo "Resources created so far:"
    if [ -n "${CLUSTER_ID:-}" ]; then echo "- Redshift Cluster: $CLUSTER_ID"; fi
    if [ -n "${ROLE_NAME:-}" ]; then echo "- IAM Role: $ROLE_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete the cluster if it exists
    if [ -n "${CLUSTER_ID:-}" ]; then
        echo "Deleting Redshift cluster: $CLUSTER_ID"
        aws redshift delete-cluster --cluster-identifier "$CLUSTER_ID" --skip-
final-cluster-snapshot 2>/dev/null || true
        echo "Waiting for cluster deletion to complete..."
        aws redshift wait cluster-deleted --cluster-identifier "$CLUSTER_ID" 2>/
dev/null || true
        echo "Cluster deleted successfully."
    fi

    # Delete the IAM role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Removing IAM role policy..."
        aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 RedshiftS3Access 2>/dev/null || true

        echo "Deleting IAM role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
```

```
    fi

    # Clean up temporary files
    rm -f redshift-trust-policy.json redshift-s3-policy.json

    echo "Cleanup completed."
}

# Trap errors and cleanup
trap 'handle_error "Script interrupted"' INT TERM

# Function to wait for SQL statement to complete
wait_for_statement() {
    local statement_id=$1
    local max_attempts=30
    local attempt=1
    local status=""

    echo "Waiting for statement $statement_id to complete..."

    while [ $attempt -le $max_attempts ]; do
        status=$(aws redshift-data describe-statement --id "$statement_id" --
query 'Status' --output text 2>/dev/null || echo "")

        if [ "$status" == "FINISHED" ]; then
            echo "Statement completed successfully."
            return 0
        elif [ "$status" == "FAILED" ]; then
            local error=$(aws redshift-data describe-statement --id
 "$statement_id" --query 'Error' --output text 2>/dev/null || echo "Unknown
 error")
            echo "Statement failed with error: $error" >&2
            return 1
        elif [ "$status" == "ABORTED" ]; then
            echo "Statement was aborted." >&2
            return 1
        fi

        echo "Statement status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 10
        ((attempt++))
    done
```

```
    echo "Timed out waiting for statement to complete." >&2
    return 1
}

# Function to check if IAM role is attached to cluster
check_role_attached() {
    local role_arn=$1
    local max_attempts=10
    local attempt=1

    echo "Checking if IAM role is attached to the cluster..."

    while [ $attempt -le $max_attempts ]; do
        local status=$(aws redshift describe-clusters \
            --cluster-identifier "$CLUSTER_ID" \
            --query "Clusters[0].IamRoles[?IamRoleArn=='$role_arn'].ApplyStatus"
 \
            --output text 2>/dev/null || echo "")

        if [ "$status" == "in-sync" ]; then
            echo "IAM role is successfully attached to the cluster."
            return 0
        fi

        echo "IAM role status: $status. Waiting... (Attempt $attempt/
$max_attempts)"
        sleep 30
        ((attempt++))
    done

    echo "Timed out waiting for IAM role to be attached." >&2
    return 1
}

# Validate required commands
for cmd in aws jq; do
    if ! command -v "$cmd" &> /dev/null; then
        handle_error "Required command '$cmd' not found. Please install it and
 try again."
    fi
done

# Validate AWS credentials
if ! aws sts get-caller-identity &>/dev/null; then
```

```
    handle_error "AWS credentials not configured or invalid"
fi

# Variables to track created resources
CLUSTER_ID="examplecluster"
ROLE_NAME="RedshiftS3Role-$(date +%s)"
DB_NAME="dev"
DB_USER="awsuser"

# Generate secure password using AWS Secrets Manager or random string
if command -v openssl &> /dev/null; then
    DB_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-20)
else
    DB_PASSWORD="TempPass$(date +%s | md5sum | cut -c1-20)"
fi

# Validate password meets requirements
if [ ${#DB_PASSWORD} -lt 8 ]; then
    handle_error "Generated password does not meet minimum length requirement"
fi

# Store password securely (optional: use AWS Secrets Manager in production)
echo "Generated database password (store securely): $DB_PASSWORD"

echo "=== Step 1: Creating Amazon Redshift Cluster ==="

# Create the Redshift cluster with encryption and audit logging enabled
echo "Creating Redshift cluster: $CLUSTER_ID"
CLUSTER_RESULT=$(aws redshift create-cluster \
  --cluster-identifier "$CLUSTER_ID" \
  --node-type ra3.xlplus \
  --number-of-nodes 2 \
  --master-username "$DB_USER" \
  --master-user-password "$DB_PASSWORD" \
  --db-name "$DB_NAME" \
  --port 5439 \
  --encrypted \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=redshift-provisioned \
  2>&1) || handle_error "Failed to create Redshift cluster"

echo "$CLUSTER_RESULT"
echo "Waiting for cluster to become available..."

# Wait for the cluster to be available
```

```
aws redshift wait cluster-available --cluster-identifier "$CLUSTER_ID" ||
 handle_error "Timeout waiting for cluster to become available"

# Get cluster status to confirm
CLUSTER_STATUS=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].ClusterStatus' \
  --output text)

echo "Cluster status: $CLUSTER_STATUS"

echo "=== Step 2: Creating IAM Role for S3 Access ==="

# Create trust policy file with restricted permissions
echo "Creating trust policy for Redshift"
cat > redshift-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "redshift.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

chmod 600 redshift-trust-policy.json

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://redshift-trust-policy.json 2>&1) ||
 handle_error "Failed to create IAM role"

echo "$ROLE_RESULT"

# Get the role ARN
ROLE_ARN=$(aws iam get-role --role-name "$ROLE_NAME" --query 'Role.Arn' --output
 text)
```

```
echo "Role ARN: $ROLE_ARN"

# Tag the IAM role
echo "Tagging IAM role: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=redshift-provisioned

# Create policy document for S3 access with principle of least privilege
echo "Creating S3 access policy"
cat > redshift-s3-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::redshift-downloads",
        "arn:aws:s3:::redshift-downloads/*"
      ]
    }
  ]
}
EOF

chmod 600 redshift-s3-policy.json

# Attach policy to role
echo "Attaching S3 access policy to role"
POLICY_RESULT=$(aws iam put-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-name RedshiftS3Access \
  --policy-document file://redshift-s3-policy.json 2>&1) || handle_error "Failed
 to attach policy to role"

echo "$POLICY_RESULT"

# Attach role to cluster
echo "Attaching IAM role to Redshift cluster"
ATTACH_ROLE_RESULT=$(aws redshift modify-cluster-iam-roles \
  --cluster-identifier "$CLUSTER_ID" \
```

```
  --add-iam-roles "$ROLE_ARN" 2>&1) || handle_error "Failed to attach role to
 cluster"

echo "$ATTACH_ROLE_RESULT"

# Wait for the role to be attached
echo "Waiting for IAM role to be attached to the cluster..."
if ! check_role_attached "$ROLE_ARN"; then
    handle_error "Failed to attach IAM role to cluster"
fi

echo "=== Step 3: Getting Cluster Connection Information ==="

# Get cluster endpoint
CLUSTER_INFO=$(aws redshift describe-clusters \
  --cluster-identifier "$CLUSTER_ID" \
  --query 'Clusters[0].Endpoint.{Address:Address,Port:Port}' \
  --output json)

echo "Cluster endpoint information:"
echo "$CLUSTER_INFO"

echo "=== Step 4: Creating Tables and Loading Data ==="

echo "Creating sales table"
SALES_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS sales; CREATE TABLE sales(salesid integer not null,
 listid integer not null distkey, sellerid integer not null, buyerid integer
 not null, eventid integer not null, dateid smallint not null sortkey, qtysold
 smallint not null, pricepaid decimal(8,2), commission decimal(8,2), saletime
 timestamp);" \
  --query 'Id' --output text)

echo "Sales table creation statement ID: $SALES_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_TABLE_ID"; then
    handle_error "Failed to create sales table"
fi

echo "Creating date table"
```

```
DATE_TABLE_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "DROP TABLE IF EXISTS date; CREATE TABLE date(dateid smallint not null
 distkey sortkey, caldate date not null, day character(3) not null, week smallint
 not null, month character(5) not null, qtr character(5) not null, year smallint
 not null, holiday boolean default('N'));" \
  --query 'Id' --output text)

echo "Date table creation statement ID: $DATE_TABLE_ID"

# Wait for statement to complete
if ! wait_for_statement "$DATE_TABLE_ID"; then
    handle_error "Failed to create date table"
fi

echo "Loading data into sales table"
SALES_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY sales FROM 's3://redshift-downloads/tickit/sales_tab.txt' DELIMITER
 '\t' TIMEFORMAT 'MM/DD/YYYY HH:MI:SS' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';"
 \
  --query 'Id' --output text)

echo "Sales data load statement ID: $SALES_LOAD_ID"

# Wait for statement to complete
if ! wait_for_statement "$SALES_LOAD_ID"; then
    handle_error "Failed to load data into sales table"
fi

echo "Loading data into date table"
DATE_LOAD_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "COPY date FROM 's3://redshift-downloads/tickit/date2008_pipe.txt'
 DELIMITER '|' REGION 'us-east-1' IAM_ROLE '$ROLE_ARN';" \
  --query 'Id' --output text)

echo "Date data load statement ID: $DATE_LOAD_ID"
```

```
# Wait for statement to complete
if ! wait_for_statement "$DATE_LOAD_ID"; then
    handle_error "Failed to load data into date table"
fi

echo "=== Step 5: Running Example Queries ==="

echo "Running query: Get definition for the sales table"
QUERY1_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT * FROM pg_table_def WHERE tablename = 'sales';" \
  --query 'Id' --output text)

echo "Query 1 statement ID: $QUERY1_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY1_ID"; then
    handle_error "Query 1 failed"
fi

# Get and display results
echo "Query 1 results (first 10 rows):"
aws redshift-data get-statement-result --id "$QUERY1_ID" --max-items 10

echo "Running query: Find total sales on a given calendar date"
QUERY2_ID=$(aws redshift-data execute-statement \
  --cluster-identifier "$CLUSTER_ID" \
  --database "$DB_NAME" \
  --db-user "$DB_USER" \
  --sql "SELECT sum(qtysold) FROM sales, date WHERE sales.dateid = date.dateid
 AND caldate = '2008-01-05';" \
  --query 'Id' --output text)

echo "Query 2 statement ID: $QUERY2_ID"

# Wait for statement to complete
if ! wait_for_statement "$QUERY2_ID"; then
    handle_error "Query 2 failed"
fi

# Get and display results
```

```
echo "Query 2 results:"
aws redshift-data get-statement-result --id "$QUERY2_ID"

echo "=== Tutorial Complete ==="
echo "The following resources were created:"
echo "- Redshift Cluster: $CLUSTER_ID"
echo "- IAM Role: $ROLE_NAME"

echo ""
echo "================================================"
echo "CLEANUP CONFIRMATION"
echo "================================================"
echo "Cleaning up all created resources..."
cleanup_resources
echo "All resources have been cleaned up."

# Securely clear password from memory
DB_PASSWORD=""

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - CreateCluster
  - CreateRole
  - DeleteCluster
  - DeleteRole
  - DeleteRolePolicy
  - DescribeClusters
  - GetRole
  - ModifyClusterIamRoles
  - PutRolePolicy
  - Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon SageMaker Feature Store 入门

以下代码示例展示了如何：

- 设置 IAM 权限
- 创建 SageMaker 执行角色
- 创建特征组
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
```

```
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"
```

```
        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2

    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
```

```
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi

    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
```

```
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi

# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"
```

```
# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
```

```
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"

CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
```

```
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
```

```
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"
```

```
CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
```

```
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"

# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
```

```
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "==========================================="
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "==========================================="
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "==========================================="
```

```
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup

- CreateRole
- DeleteBucket
- DeleteFeatureGroup
- DeleteRole
- DescribeFeatureGroup
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListFeatureGroups
- PutPublicAccessBlock
- Rm

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Config 入门

以下代码示例展示了如何：

- 创建 Amazon S3 存储桶
- 创建 Amazon SNS 主题
- 为 Config 创建 IAM 角色
- 设置 Config 配置记录器
- 设置 Config 传输通道
- 启动配置记录器
- 验证 Config 设置

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
```

```
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
```

```
    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
    echo "CREATED RESOURCES"
    echo "============================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "============================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
```

```
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Step Functions 入门

以下代码示例展示了如何：

- 为 Step Functions 创建 IAM 角色
- 创建第一个状态机
- 启动状态机执行
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS Step Functions Getting Started Tutorial Script
# This script creates and runs a Step Functions state machine based on the AWS
 Step Functions Getting Started tutorial

set -euo pipefail

# Security: Restrict umask to prevent unintended file permissions
umask 077

# Parse command line arguments
AUTO_CLEANUP=true
while [[ $# -gt 0 ]]; do
    case $1 in
        --auto-cleanup)
            AUTO_CLEANUP=true
            shift
            ;;
        -h|--help)
            echo "Usage: $0 [--auto-cleanup] [--help]"
```

```
            echo "  --auto-cleanup: Automatically clean up resources without
 prompting"
            echo "  --help: Show this help message"
            exit 0
            ;;
        *)
            echo "Unknown option: $1"
            echo "Use --help for usage information"
            exit 1
            ;;
    esac
done

# Set up logging with secure permissions
LOG_FILE="step-functions-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"

# Security: Use process substitution with explicit FD cleanup
exec 3>&1 4>&2
exec > >(tee -a "$LOG_FILE") 2>&1
trap 'exec 1>&3 2>&4 3>&- 4>&-' EXIT

echo "Starting AWS Step Functions Getting Started Tutorial..."
echo "Logging to $LOG_FILE"

# Verify AWS CLI is installed and configured
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed"
    exit 1
fi

# Verify AWS credentials are configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS credentials are not configured or invalid"
    exit 1
fi

# Check if jq is available for better JSON parsing
if ! command -v jq &> /dev/null; then
    echo "WARNING: jq is not installed. Using basic JSON parsing which may be
 less reliable."
    echo "Consider installing jq for better error handling: brew install jq
 (macOS) or apt-get install jq (Ubuntu)"
```

```
    USE_JQ=false
else
    USE_JQ=true
fi

# Use fixed region that supports Amazon Comprehend
CURRENT_REGION="us-west-2"
echo "Using fixed AWS region: $CURRENT_REGION (supports Amazon Comprehend)"

# Set AWS CLI to use the fixed region for all commands
export AWS_DEFAULT_REGION="$CURRENT_REGION"
export AWS_REGION="$CURRENT_REGION"

# Amazon Comprehend is available in us-west-2, so we can always enable it
echo "Amazon Comprehend is available in region $CURRENT_REGION"
SKIP_COMPREHEND=false

# Security: Initialize all resource variables
STATE_MACHINE_ARN=""
ROLE_NAME=""
ROLE_ARN=""
POLICY_ARN=""
STEPFUNCTIONS_POLICY_ARN=""
EXECUTION_ARN=""
EXECUTION2_ARN=""
EXECUTION3_ARN=""

# Performance: Cache for AWS API calls to reduce redundant requests
declare -A API_CACHE

# Function to make cached AWS CLI calls
aws_call_cached() {
    local cache_key="$1"
    shift

    if [[ -v API_CACHE["$cache_key"] ]]; then
        echo "${API_CACHE[$cache_key]}"
        return 0
    fi

    local result
    result=$(aws "$@" 2>&1) || return $?
    API_CACHE["$cache_key"]="$result"
    echo "$result"
```

```
}

# Function to check for API errors in JSON response with optimized jq usage
check_api_error() {
    local response="$1"
    local operation="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        # Use jq for more reliable JSON parsing with efficient error detection
        if echo "$response" | jq -e '.Error // .error // empty' > /dev/null 2>&1;
 then
            local error_message=$(echo "$response" | jq -r
 '.Error.Message // .Error.Code // .error // "Unknown error"' 2>/dev/null)
            handle_error "$operation failed: $error_message"
        fi
    else
        # Fallback to grep-based detection with optimized pattern
        if echo "$response" | grep -qE '"[Ee]rror":|"error":'; then
            handle_error "$operation failed: $response"
        fi
    fi
}

# Function to extract JSON field efficiently
extract_json_field() {
    local json="$1"
    local field="$2"

    if [[ "$USE_JQ" == "true" ]]; then
        echo "$json" | jq -r "$field" 2>/dev/null
    else
        echo "$json" | grep -oP "\"${field}\":\s*\"\K[^\"]+|\"${field}\":\s*
\K[^,}]+" | head -1
    fi
}

# Function to securely wait for resource propagation with exponential backoff
wait_for_propagation() {
    local resource_type="$1"
    local wait_time="${2:-10}"

    # Validate wait_time is a positive integer
    if ! [[ "$wait_time" =~ ^[0-9]+$ ]] || [ "$wait_time" -lt 1 ] ||
 [ "$wait_time" -gt 300 ]; then
```

```
        echo "WARNING: Invalid wait time $wait_time, using default 10 seconds"
        wait_time=10
    fi

    echo "Waiting for $resource_type to propagate ($wait_time seconds)..."
    sleep "$wait_time"
}

# Function to validate JSON file efficiently
validate_json_file() {
    local file="$1"

    if [[ "$USE_JQ" == "true" ]]; then
        if ! jq empty "$file" 2>/dev/null; then
            handle_error "Invalid JSON in $file"
        fi
    fi
}

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created:"
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "- State Machine: $STATE_MACHINE_ARN"
    fi
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "- IAM Role: $ROLE_NAME"
    fi
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "- IAM Policy: $POLICY_ARN"
    fi
    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "- Step Functions Policy: $STEPFUNCTIONS_POLICY_ARN"
    fi

    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to securely clean up resources with parallel deletion
cleanup() {
    echo "Cleaning up resources..."
```

```
    # Delete state machine if it exists
    if [ -n "${STATE_MACHINE_ARN:-}" ]; then
        echo "Deleting state machine: $STATE_MACHINE_ARN"
        aws stepfunctions delete-state-machine --state-machine-arn
 "$STATE_MACHINE_ARN" 2>/dev/null &
    fi

    # Detach and delete policies if they exist
    if [ -n "${POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Comprehend policy $POLICY_ARN from role $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ] && [ -n "${ROLE_NAME:-}" ]; then
        echo "Detaching Step Functions policy $STEPFUNCTIONS_POLICY_ARN from role
 $ROLE_NAME"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/null &
    fi

    # Wait for detach operations to complete
    wait 2>/dev/null || true

    # Delete custom policies if they exist
    if [ -n "${POLICY_ARN:-}" ]; then
        echo "Deleting Comprehend policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" 2>/dev/null &
    fi

    if [ -n "${STEPFUNCTIONS_POLICY_ARN:-}" ]; then
        echo "Deleting Step Functions policy: $STEPFUNCTIONS_POLICY_ARN"
        aws iam delete-policy --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>/dev/
null &
    fi

    # Wait for policy deletion to complete
    wait 2>/dev/null || true

    # Delete role if it exists
    if [ -n "${ROLE_NAME:-}" ]; then
        echo "Deleting role: $ROLE_NAME"
```

```
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || echo "Failed
 to delete role"
    fi

    # Remove temporary files securely
    echo "Removing temporary files"
    local temp_files=(
        "hello-world.json"
        "updated-hello-world.json"
        "sentiment-hello-world.json"
        "step-functions-trust-policy.json"
        "comprehend-policy.json"
        "stepfunctions-policy.json"
        "input.json"
        "sentiment-input.json"
    )

    for file in "${temp_files[@]}"; do
        if [ -f "$file" ]; then
            if command -v shred &> /dev/null; then
                shred -vfz -n 3 "$file" 2>/dev/null || rm -f "$file"
            else
                rm -f "$file"
            fi
        fi
    done
}

# Security: Set trap to cleanup on script exit
trap cleanup EXIT

# Generate a secure random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
ROLE_NAME="StepFunctionsHelloWorldRole-${RANDOM_ID}"
POLICY_NAME="DetectSentimentPolicy-${RANDOM_ID}"
STATE_MACHINE_NAME="MyFirstStateMachine-${RANDOM_ID}"

echo "Using random identifier: $RANDOM_ID"
echo "Role name: $ROLE_NAME"
echo "Policy name: $POLICY_NAME"
echo "State machine name: $STATE_MACHINE_NAME"

# Step 1: Create the state machine definition
echo "Creating state machine definition..."
```

```
cat > hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Result": {
        "IsHelloWorldExample": true,
        "ExecutionWaitTimeInSeconds": 10
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
```

```
        },
        {
          "StartAt": "Process2",
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Create IAM role trust policy
echo "Creating IAM role trust policy..."
cat > step-functions-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
```

```
        "Service": "states.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://step-functions-trust-policy.json 2>&1)

check_api_error "$ROLE_RESULT" "Create IAM role"
echo "Role created successfully"
aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-step-functions-gs

# Get the role ARN
ROLE_ARN=$(extract_json_field "$ROLE_RESULT" ".Role.Arn")

if [ -z "$ROLE_ARN" ]; then
    handle_error "Failed to extract role ARN"
fi
echo "Role ARN: $ROLE_ARN"

# Create a custom policy for Step Functions with least privilege
echo "Creating custom policy for Step Functions..."
cat > stepfunctions-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "states:StartExecution",
        "states:DescribeExecution",
        "states:StopExecution"
      ],
      "Resource": "arn:aws:states:*:*:stateMachine:*"
    }
  ]
}
```

```
EOF

# Create the policy
echo "Creating Step Functions policy..."
STEPFUNCTIONS_POLICY_RESULT=$(aws iam create-policy \
  --policy-name "StepFunctionsPolicy-${RANDOM_ID}" \
  --policy-document file://stepfunctions-policy.json 2>&1)

check_api_error "$STEPFUNCTIONS_POLICY_RESULT" "Create Step Functions policy"
echo "Step Functions policy created successfully"

# Get the policy ARN
STEPFUNCTIONS_POLICY_ARN=$(extract_json_field "$STEPFUNCTIONS_POLICY_RESULT"
 ".Policy.Arn")

if [ -z "$STEPFUNCTIONS_POLICY_ARN" ]; then
    handle_error "Failed to extract Step Functions policy ARN"
fi
echo "Step Functions policy ARN: $STEPFUNCTIONS_POLICY_ARN"

# Attach policy to the role
echo "Attaching Step Functions policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$STEPFUNCTIONS_POLICY_ARN" 2>&1)

if [ $? -ne 0 ]; then
    handle_error "Failed to attach Step Functions policy to role: $ATTACH_RESULT"
fi

# Wait for role to propagate (IAM changes can take time to propagate)
wait_for_propagation "IAM role" 8

# Create state machine
echo "Creating state machine: $STATE_MACHINE_NAME"
SM_RESULT=$(aws stepfunctions create-state-machine \
  --name "$STATE_MACHINE_NAME" \
  --definition file://hello-world.json \
  --role-arn "$ROLE_ARN" \
  --type STANDARD \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-step-functions-gs
 2>&1)

check_api_error "$SM_RESULT" "Create state machine"
```

```
echo "State machine created successfully"

# Get the state machine ARN
STATE_MACHINE_ARN=$(extract_json_field "$SM_RESULT" ".stateMachineArn")

if [ -z "$STATE_MACHINE_ARN" ]; then
    handle_error "Failed to extract state machine ARN"
fi
echo "State machine ARN: $STATE_MACHINE_ARN"

# Step 2: Start the state machine execution
echo "Starting state machine execution..."
EXEC_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello001-${RANDOM_ID}" 2>&1)

check_api_error "$EXEC_RESULT" "Start execution"
echo "Execution started successfully"

# Get the execution ARN
EXECUTION_ARN=$(extract_json_field "$EXEC_RESULT" ".executionArn")

if [ -z "$EXECUTION_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION_ARN"

# Wait for execution to complete (the workflow has a 10-second wait state)
echo "Waiting for execution to complete (12 seconds)..."
sleep 12

# Check execution status
echo "Checking execution status..."
EXEC_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION_ARN" 2>&1)

echo "Execution status: $EXEC_STATUS"

# Step 3: Update state machine to process external input
echo "Updating state machine to process external input..."
cat > updated-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example of the Amazon States Language using a Pass
 state",
```

```
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "FailState"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
          "States": {
```

```
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    },
    "FailState": {
      "Type": "Fail",
      "Error": "NotHelloWorldExample",
      "Cause": "The IsHelloWorldExample value was false"
    }
  }
}
EOF

# Update state machine
echo "Updating state machine..."
UPDATE_RESULT=$(aws stepfunctions update-state-machine \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --definition file://updated-hello-world.json \
  --role-arn "$ROLE_ARN" 2>&1)

check_api_error "$UPDATE_RESULT" "Update state machine"
echo "State machine updated successfully"

# Create input file with strict validation
echo "Creating input file..."
cat > input.json << 'EOF'
```

```
{
  "wait": 5,
  "hello_world": true
}
EOF

# Validate input JSON
validate_json_file "input.json"

# Start execution with input
echo "Starting execution with input..."
EXEC2_RESULT=$(aws stepfunctions start-execution \
  --state-machine-arn "$STATE_MACHINE_ARN" \
  --name "hello002-${RANDOM_ID}" \
  --input file://input.json 2>&1)

check_api_error "$EXEC2_RESULT" "Start execution with input"
echo "Execution with input started successfully"

# Get the execution ARN
EXECUTION2_ARN=$(extract_json_field "$EXEC2_RESULT" ".executionArn")

if [ -z "$EXECUTION2_ARN" ]; then
    handle_error "Failed to extract execution ARN"
fi
echo "Execution ARN: $EXECUTION2_ARN"

# Wait for execution to complete (the workflow has a 5-second wait state)
echo "Waiting for execution to complete (8 seconds)..."
sleep 8

# Check execution status
echo "Checking execution status..."
EXEC2_STATUS=$(aws stepfunctions describe-execution \
  --execution-arn "$EXECUTION2_ARN" 2>&1)

echo "Execution status: $EXEC2_STATUS"

# Step 4: Integrate Amazon Comprehend for sentiment analysis (if available)
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Creating policy for Amazon Comprehend access with least privilege..."
    cat > comprehend-policy.json << 'EOF'
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "comprehend:DetectSentiment"
      ],
      "Resource": "*"
    }
  ]
}
EOF

    # Create policy
    echo "Creating IAM policy: $POLICY_NAME"
    POLICY_RESULT=$(aws iam create-policy \
      --policy-name "$POLICY_NAME" \
      --policy-document file://comprehend-policy.json 2>&1)

    check_api_error "$POLICY_RESULT" "Create Comprehend policy"
    echo "Comprehend policy created successfully"

    # Get policy ARN
    POLICY_ARN=$(extract_json_field "$POLICY_RESULT" ".Policy.Arn")

    if [ -z "$POLICY_ARN" ]; then
        handle_error "Failed to extract policy ARN"
    fi
    echo "Policy ARN: $POLICY_ARN"

    # Attach policy to role
    echo "Attaching policy to role..."
    ATTACH2_RESULT=$(aws iam attach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN" 2>&1)

    if [ $? -ne 0 ]; then
        handle_error "Failed to attach policy to role: $ATTACH2_RESULT"
    fi

    # Create updated state machine definition with sentiment analysis
    echo "Creating updated state machine definition with sentiment analysis..."
    cat > sentiment-hello-world.json << 'EOF'
{
  "Comment": "A Hello World example with sentiment analysis",
```

```
  "StartAt": "SetVariables",
  "States": {
    "SetVariables": {
      "Type": "Pass",
      "Parameters": {
        "IsHelloWorldExample.$": "$.hello_world",
        "ExecutionWaitTimeInSeconds.$": "$.wait",
        "FeedbackComment.$": "$.feedback_comment"
      },
      "Next": "IsHelloWorldExample"
    },
    "IsHelloWorldExample": {
      "Type": "Choice",
      "Choices": [
        {
          "Variable": "$.IsHelloWorldExample",
          "BooleanEquals": true,
          "Next": "WaitState"
        }
      ],
      "Default": "DetectSentiment"
    },
    "WaitState": {
      "Type": "Wait",
      "SecondsPath": "$.ExecutionWaitTimeInSeconds",
      "Next": "ParallelProcessing"
    },
    "ParallelProcessing": {
      "Type": "Parallel",
      "Branches": [
        {
          "StartAt": "Process1",
          "States": {
            "Process1": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 1"
              },
              "End": true
            }
          }
        },
        {
          "StartAt": "Process2",
```

```
          "States": {
            "Process2": {
              "Type": "Pass",
              "Result": {
                "message": "Processing task 2"
              },
              "End": true
            }
          }
        }
      ],
      "Next": "CheckpointState"
    },
    "CheckpointState": {
      "Type": "Pass",
      "Result": {
        "CheckpointMessage": "Workflow completed successfully!"
      },
      "Next": "SuccessState"
    },
    "DetectSentiment": {
      "Type": "Task",
      "Resource": "arn:aws:states:::aws-sdk:comprehend:detectSentiment",
      "Parameters": {
        "LanguageCode": "en",
        "Text.$": "$.FeedbackComment"
      },
      "Next": "SuccessState"
    },
    "SuccessState": {
      "Type": "Succeed"
    }
  }
}
EOF

    # Validate sentiment state machine JSON
    validate_json_file "sentiment-hello-world.json"

    # Wait for IAM changes to propagate
    wait_for_propagation "IAM changes" 8

    # Update state machine
    echo "Updating state machine with sentiment analysis..."
```

```
    UPDATE2_RESULT=$(aws stepfunctions update-state-machine \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --definition file://sentiment-hello-world.json \
      --role-arn "$ROLE_ARN" 2>&1)

    check_api_error "$UPDATE2_RESULT" "Update state machine with sentiment
 analysis"
    echo "State machine updated with sentiment analysis successfully"

    # Create input file with feedback comment
    echo "Creating input file with feedback comment..."
    cat > sentiment-input.json << 'EOF'
{
  "hello_world": false,
  "wait": 5,
  "feedback_comment": "This getting started with Step Functions workshop is a
 challenge!"
}
EOF

    # Validate sentiment input JSON
    validate_json_file "sentiment-input.json"

    # Start execution with sentiment analysis input
    echo "Starting execution with sentiment analysis input..."
    EXEC3_RESULT=$(aws stepfunctions start-execution \
      --state-machine-arn "$STATE_MACHINE_ARN" \
      --name "hello003-${RANDOM_ID}" \
      --input file://sentiment-input.json 2>&1)

    check_api_error "$EXEC3_RESULT" "Start execution with sentiment analysis"
    echo "Execution with sentiment analysis started successfully"

    # Get the execution ARN
    EXECUTION3_ARN=$(extract_json_field "$EXEC3_RESULT" ".executionArn")

    if [ -z "$EXECUTION3_ARN" ]; then
        handle_error "Failed to extract execution ARN"
    fi
    echo "Execution ARN: $EXECUTION3_ARN"

    # Wait for execution to complete
    echo "Waiting for execution to complete (3 seconds)..."
    sleep 3
```

```
    # Check execution status
    echo "Checking execution status..."
    EXEC3_STATUS=$(aws stepfunctions describe-execution \
      --execution-arn "$EXECUTION3_ARN" 2>&1)

    echo "Execution status: $EXEC3_STATUS"
else
    echo "Skipping Amazon Comprehend integration (not available in
 $CURRENT_REGION)"
    EXECUTION3_ARN=""
fi

# Display summary of resources created
echo ""
echo "========================================="
echo "RESOURCES CREATED"
echo "========================================="
echo "State Machine: $STATE_MACHINE_ARN"
echo "IAM Role: $ROLE_NAME"
echo "Step Functions Policy: StepFunctionsPolicy-${RANDOM_ID}
 ($STEPFUNCTIONS_POLICY_ARN)"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "Comprehend Policy: $POLICY_NAME ($POLICY_ARN)"
fi
echo "Executions:"
echo "  - hello001-${RANDOM_ID}: $EXECUTION_ARN"
echo "  - hello002-${RANDOM_ID}: $EXECUTION2_ARN"
if [[ "$SKIP_COMPREHEND" == "false" ]]; then
    echo "  - hello003-${RANDOM_ID}: $EXECUTION3_ARN"
fi
echo "========================================="

# Cleanup
echo ""
echo "========================================="
echo "CLEANUP"
echo "========================================="
echo "Auto-cleanup enabled. Cleaning up resources..."

echo "All resources have been cleaned up."

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateStateMachine
  - DeletePolicy
  - DeleteRole
  - DeleteStateMachine
  - DescribeExecution
  - DetachRolePolicy
  - StartExecution
  - UpdateStateMachine

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包管理 IAM 访问密钥

以下代码示例显示如何管理您的访问密钥。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建并列出访问密钥。
- 了解上次使用访问密钥的时间和方式。
- 更新和删除访问密钥。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 访问密钥操作的函数。

```
import logging
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)

iam = boto3.resource("iam")

def list_keys(user_name):
    """
    Lists the keys owned by the specified user.

    :param user_name: The name of the user.
    :return: The list of keys owned by the user.
    """
    try:
        keys = list(iam.User(user_name).access_keys.all())
        logger.info("Got %s access keys for %s.", len(keys), user_name)
    except ClientError:
        logger.exception("Couldn't get access keys for %s.", user_name)
        raise
    else:
        return keys



def create_key(user_name):
    """
    Creates an access key for the specified user. Each user can have a
    maximum of two keys.
```

```
    :param user_name: The name of the user.
    :return: The created access key.
    """
    try:
        key_pair = iam.User(user_name).create_access_key_pair()
        logger.info(
            "Created access key pair for %s. Key ID is %s.",
            key_pair.user_name,
            key_pair.id,
        )
    except ClientError:
        logger.exception("Couldn't create access key pair for %s.", user_name)
        raise
    else:
        return key_pair



def get_last_use(key_id):
    """
    Gets information about when and how a key was last used.

    :param key_id: The ID of the key to look up.
    :return: Information about the key's last use.
    """
    try:
        response = iam.meta.client.get_access_key_last_used(AccessKeyId=key_id)
        last_used_date = response["AccessKeyLastUsed"].get("LastUsedDate", None)
        last_service = response["AccessKeyLastUsed"].get("ServiceName", None)
        logger.info(
            "Key %s was last used by %s on %s to access %s.",
            key_id,
            response["UserName"],
            last_used_date,
            last_service,
        )
    except ClientError:
        logger.exception("Couldn't get last use of key %s.", key_id)
        raise
    else:
        return response
```

```
def update_key(user_name, key_id, activate):
    """
    Updates the status of a key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to update.
    :param activate: When True, the key is activated. Otherwise, the key is
 deactivated.
    """

    try:
        key = iam.User(user_name).AccessKey(key_id)
        if activate:
            key.activate()
        else:
            key.deactivate()
        logger.info("%s key %s.", "Activated" if activate else "Deactivated",
 key_id)
    except ClientError:
        logger.exception(
            "Couldn't %s key %s.", "Activate" if activate else "Deactivate",
 key_id
        )
        raise



def delete_key(user_name, key_id):
    """
    Deletes a user's access key.

    :param user_name: The user that owns the key.
    :param key_id: The ID of the key to delete.
    """

    try:
        key = iam.AccessKey(user_name, key_id)
        key.delete()
        logger.info("Deleted access key %s for %s.", key.id, key.user_name)
    except ClientError:
        logger.exception("Couldn't delete key %s for %s", key_id, user_name)
        raise
```

使用包装器函数为当前用户执行访问密钥操作。

```
def usage_demo():
    """Shows how to create and manage access keys."""

    def print_keys():
        """Gets and prints the current keys for a user."""
        current_keys = list_keys(current_user_name)
        print("The current user's keys are now:")
        print(*[f"{key.id}: {key.status}" for key in current_keys], sep="\n")

    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management access key demo.")
    print("-" * 88)
    current_user_name = iam.CurrentUser().user_name
    print(
        f"This demo creates an access key for the current user "
        f"({current_user_name}), manipulates the key in a few ways, and then "
        f"deletes it."
    )
    all_keys = list_keys(current_user_name)
    if len(all_keys) == 2:
        print(
            "The current user already has the maximum of 2 access keys. To run "
            "this demo, either delete one of the access keys or use a user "
            "that has only 1 access key."
        )
    else:
        new_key = create_key(current_user_name)
        print(f"Created a new key with id {new_key.id} and secret
 {new_key.secret}.")
        print_keys()
        existing_key = next(key for key in all_keys if key != new_key)
        last_use = get_last_use(existing_key.id)["AccessKeyLastUsed"]
        print(
            f"Key {all_keys[0].id} was last used to access
 {last_use['ServiceName']} "
            f"on {last_use['LastUsedDate']}"
        )
        update_key(current_user_name, new_key.id, False)
```

```
        print(f"Key {new_key.id} is now deactivated.")
        print_keys()
        delete_key(current_user_name, new_key.id)
        print_keys()
        print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateAccessKey
  - DeleteAccessKey
  - GetAccessKeyLastUsed
  - ListAccessKeys
  - UpdateAccessKey

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包管理 IAM policy

以下代码示例展示了如何：

- 创建和列出策略。
- 创建和获取策略版本。
- 将策略回滚到以前的版本。
- 删除策略。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM policy 操作的函数。

```
import json
import logging
import operator
import pprint
import time

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_policy(name, description, actions, resource_arn):
    """
    Creates a policy that contains a single statement.

    :param name: The name of the policy to create.
    :param description: The description of the policy.
    :param actions: The actions allowed by the policy. These typically take the
                    form of service:action, such as s3:PutObject.
    :param resource_arn: The Amazon Resource Name (ARN) of the resource this
 policy
                         applies to. This ARN can contain wildcards, such as
                         'arn:aws:s3:::amzn-s3-demo-bucket/*' to allow actions on
 all objects
                         in the bucket named 'amzn-s3-demo-bucket'.
    :return: The newly created policy.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
    try:
        policy = iam.create_policy(
            PolicyName=name,
            Description=description,
            PolicyDocument=json.dumps(policy_doc),
        )
        logger.info("Created policy %s.", policy.arn)
    except ClientError:
        logger.exception("Couldn't create policy %s.", name)
```

```
        raise
    else:
        return policy



def list_policies(scope):
    """
    Lists the policies in the current account.

    :param scope: Limits the kinds of policies that are returned. For example,
                  'Local' specifies that only locally managed policies are
 returned.
    :return: The list of policies.
    """
    try:
        policies = list(iam.policies.filter(Scope=scope))
        logger.info("Got %s policies in scope '%s'.", len(policies), scope)
    except ClientError:
        logger.exception("Couldn't get policies for scope '%s'.", scope)
        raise
    else:
        return policies



def create_policy_version(policy_arn, actions, resource_arn, set_as_default):
    """
    Creates a policy version. Policies can have up to five versions. The default
    version is the one that is used for all resources that reference the policy.

    :param policy_arn: The ARN of the policy.
    :param actions: The actions to allow in the policy version.
    :param resource_arn: The ARN of the resource this policy version applies to.
    :param set_as_default: When True, this policy version is set as the default
                           version for the policy. Otherwise, the default
                           is not changed.
    :return: The newly created policy version.
    """
    policy_doc = {
        "Version":"2012-10-17",
        "Statement": [{"Effect": "Allow", "Action": actions, "Resource":
 resource_arn}],
    }
```

```
    try:
        policy = iam.Policy(policy_arn)
        policy_version = policy.create_version(
            PolicyDocument=json.dumps(policy_doc), SetAsDefault=set_as_default
        )
        logger.info(
            "Created policy version %s for policy %s.",
            policy_version.version_id,
            policy_version.arn,
        )
    except ClientError:
        logger.exception("Couldn't create a policy version for %s.", policy_arn)
        raise
    else:
        return policy_version



def get_default_policy_statement(policy_arn):
    """
    Gets the statement of the default version of the specified policy.

    :param policy_arn: The ARN of the policy to look up.
    :return: The statement of the default policy version.
    """
    try:
        policy = iam.Policy(policy_arn)
        # To get an attribute of a policy, the SDK first calls get_policy.
        policy_doc = policy.default_version.document
        policy_statement = policy_doc.get("Statement", None)
        logger.info("Got default policy doc for %s.", policy.policy_name)
        logger.info(policy_doc)
    except ClientError:
        logger.exception("Couldn't get default policy statement for %s.",
 policy_arn)
        raise
    else:
        return policy_statement



def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.
```

```
    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
```

```
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version



def delete_policy(policy_arn):
    """
    Deletes a policy.

    :param policy_arn: The ARN of the policy to delete.
    """
    try:
        iam.Policy(policy_arn).delete()
        logger.info("Deleted policy %s.", policy_arn)
    except ClientError:
        logger.exception("Couldn't delete policy %s.", policy_arn)
        raise
```

使用包装器函数来创建策略、更新版本并获取相关信息。

```
def usage_demo():
    """Shows how to use the policy functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management policy demo.")
    print("-" * 88)
    print(
        "Policies let you define sets of permissions that can be attached to "
        "other IAM resources, like users and roles."
    )
    bucket_arn = f"arn:aws:s3:::amzn-s3-demo-bucket"
    policy = create_policy(
        "demo-iam-policy",
        "Policy for IAM demonstration.",
        ["s3:ListObjects"],
        bucket_arn,
    )
```

```
    print(f"Created policy {policy.policy_name}.")
    policies = list_policies("Local")
    print(f"Your account has {len(policies)} managed policies:")
    print(*[pol.policy_name for pol in policies], sep=", ")
    time.sleep(1)
    policy_version = create_policy_version(
        policy.arn, ["s3:PutObject"], bucket_arn, True
    )
    print(
        f"Added policy version {policy_version.version_id} to policy "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is:")
    pprint.pprint(default_statement)
    rollback_version = rollback_policy_version(policy.arn)
    print(
        f"Rolled back to version {rollback_version.version_id} for "
        f"{policy.policy_name}."
    )
    default_statement = get_default_policy_statement(policy.arn)
    print(f"The default policy statement for {policy.policy_name} is now:")
    pprint.pprint(default_statement)
    delete_policy(policy.arn)
    print(f"Deleted policy {policy.policy_name}.")
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreatePolicy
  - CreatePolicyVersion
  - DeletePolicy
  - DeletePolicyVersion
  - GetPolicyVersion
  - ListPolicies
  - ListPolicyVersions
  - SetDefaultPolicyVersion

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包管理 IAM 角色

以下代码示例展示了如何：

- 创建 IAM 角色。
- 为角色附加和分离策略。
- 删除角色。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 角色操作的函数。

```
import json
import logging
import pprint

import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def create_role(role_name, allowed_services):
    """
    Creates a role that lets a list of specified services assume the role.

    :param role_name: The name of the role.
    :param allowed_services: The services that can assume the role.
    :return: The newly created role.
    """
```

```
    trust_policy = {
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {"Service": service},
                "Action": "sts:AssumeRole",
            }
            for service in allowed_services
        ],
    }

    try:
        role = iam.create_role(
            RoleName=role_name, AssumeRolePolicyDocument=json.dumps(trust_policy)
        )
        logger.info("Created role %s.", role.name)
    except ClientError:
        logger.exception("Couldn't create role %s.", role_name)
        raise
    else:
        return role



def attach_policy(role_name, policy_arn):
    """
    Attaches a policy to a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).attach_policy(PolicyArn=policy_arn)
        logger.info("Attached policy %s to role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception("Couldn't attach policy %s to role %s.", policy_arn,
 role_name)
        raise



def detach_policy(role_name, policy_arn):
```

```
    """
    Detaches a policy from a role.

    :param role_name: The name of the role. **Note** this is the name, not the
 ARN.
    :param policy_arn: The ARN of the policy.
    """
    try:
        iam.Role(role_name).detach_policy(PolicyArn=policy_arn)
        logger.info("Detached policy %s from role %s.", policy_arn, role_name)
    except ClientError:
        logger.exception(
            "Couldn't detach policy %s from role %s.", policy_arn, role_name
        )
        raise



def delete_role(role_name):
    """
    Deletes a role.

    :param role_name: The name of the role to delete.
    """
    try:
        iam.Role(role_name).delete()
        logger.info("Deleted role %s.", role_name)
    except ClientError:
        logger.exception("Couldn't delete role %s.", role_name)
        raise
```

使用包装器函数创建角色，然后附加和分离策略。

```
def usage_demo():
    """Shows how to use the role functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
    print("Welcome to the AWS Identity and Account Management role demo.")
    print("-" * 88)
    print(
```

```
        "Roles let you define sets of permissions and can be assumed by "
        "other entities, like users and services."
    )
    print("The first 10 roles currently in your account are:")
    roles = list_roles(10)
    print(f"The inline policies for role {roles[0].name} are:")
    list_policies(roles[0].name)
    role = create_role(
        "demo-iam-role", ["lambda.amazonaws.com",
"batchoperations.s3.amazonaws.com"]
    )
    print(f"Created role {role.name}, with trust policy:")
    pprint.pprint(role.assume_role_policy_document)
    policy_arn = "arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess"
    attach_policy(role.name, policy_arn)
    print(f"Attached policy {policy_arn} to {role.name}.")
    print(f"Policies attached to role {role.name} are:")
    list_attached_policies(role.name)
    detach_policy(role.name, policy_arn)
    print(f"Detached policy {policy_arn} from {role.name}.")
    delete_role(role.name)
    print(f"Deleted {role.name}.")
    print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - AttachRolePolicy
  - CreateRole
  - DeleteRole
  - DetachRolePolicy

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包管理您的 IAM 账户

以下代码示例展示了如何：

- 获取并更新账户别名。
- 生成用户和凭证的报告。
- 获取账户使用情况的摘要。
- 获取您的账户中所有用户、组、角色和策略的详细信息，包括它们之间的相互关系。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建包装 IAM 账户操作的函数。

```
import logging
import pprint
import sys
import time
import boto3
from botocore.exceptions import ClientError

logger = logging.getLogger(__name__)
iam = boto3.resource("iam")

def list_aliases():
    """
    Gets the list of aliases for the current account. An account has at most one
 alias.

    :return: The list of aliases for the account.
    """
    try:
        response = iam.meta.client.list_account_aliases()
        aliases = response["AccountAliases"]
        if len(aliases) > 0:
            logger.info("Got aliases for your account: %s.", ",".join(aliases))
        else:
            logger.info("Got no aliases for your account.")
```

```
    except ClientError:
        logger.exception("Couldn't list aliases for your account.")
        raise
    else:
        return response["AccountAliases"]



def create_alias(alias):
    """
    Creates an alias for the current account. The alias can be used in place of
 the
    account ID in the sign-in URL. An account can have only one alias. When a new
    alias is created, it replaces any existing alias.

    :param alias: The alias to assign to the account.
    """

    try:
        iam.create_account_alias(AccountAlias=alias)
        logger.info("Created an alias '%s' for your account.", alias)
    except ClientError:
        logger.exception("Couldn't create alias '%s' for your account.", alias)
        raise



def delete_alias(alias):
    """
    Removes the alias from the current account.

    :param alias: The alias to remove.
    """
    try:
        iam.meta.client.delete_account_alias(AccountAlias=alias)
        logger.info("Removed alias '%s' from your account.", alias)
    except ClientError:
        logger.exception("Couldn't remove alias '%s' from your account.", alias)
        raise



def generate_credential_report():
    """
```

```
    Starts generation of a credentials report about the current account. After
    calling this function to generate the report, call get_credential_report
    to get the latest report. A new report can be generated a minimum of four
 hours
    after the last one was generated.
    """
    try:
        response = iam.meta.client.generate_credential_report()
        logger.info(
            "Generating credentials report for your account. " "Current state is
 %s.",
            response["State"],
        )
    except ClientError:
        logger.exception("Couldn't generate a credentials report for your
 account.")
        raise
    else:
        return response



def get_credential_report():
    """
    Gets the most recently generated credentials report about the current
 account.

    :return: The credentials report.
    """
    try:
        response = iam.meta.client.get_credential_report()
        logger.debug(response["Content"])
    except ClientError:
        logger.exception("Couldn't get credentials report.")
        raise
    else:
        return response["Content"]



def get_summary():
    """
    Gets a summary of account usage.
```

```
    :return: The summary of account usage.
    """
    try:
        summary = iam.AccountSummary()
        logger.debug(summary.summary_map)
    except ClientError:
        logger.exception("Couldn't get a summary for your account.")
        raise
    else:
        return summary.summary_map



def get_authorization_details(response_filter):
    """
    Gets an authorization detail report for the current account.

    :param response_filter: A list of resource types to include in the report,
 such
                            as users or roles. When not specified, all resources
                            are included.
    :return: The authorization detail report.
    """
    try:
        account_details = iam.meta.client.get_account_authorization_details(
            Filter=response_filter
        )
        logger.debug(account_details)
    except ClientError:
        logger.exception("Couldn't get details for your account.")
        raise
    else:
        return account_details
```

调用 wrapper 函数来更改账户别名并获取有关账户的报告。

```
def usage_demo():
    """Shows how to use the account functions."""
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")
    print("-" * 88)
```

```
    print("Welcome to the AWS Identity and Account Management account demo.")
    print("-" * 88)
    print(
        "Setting an account alias lets you use the alias in your sign-in URL "
        "instead of your account number."
    )
    old_aliases = list_aliases()
    if len(old_aliases) > 0:
        print(f"Your account currently uses '{old_aliases[0]}' as its alias.")
    else:
        print("Your account currently has no alias.")
    for index in range(1, 3):
        new_alias = f"alias-{index}-{time.time_ns()}"
        print(f"Setting your account alias to {new_alias}")
        create_alias(new_alias)
    current_aliases = list_aliases()
    print(f"Your account alias is now {current_aliases}.")
    delete_alias(current_aliases[0])
    print(f"Your account now has no alias.")
    if len(old_aliases) > 0:
        print(f"Restoring your original alias back to {old_aliases[0]}...")
        create_alias(old_aliases[0])

    print("-" * 88)
    print("You can get various reports about your account.")
    print("Let's generate a credentials report...")
    report_state = None
    while report_state != "COMPLETE":
        cred_report_response = generate_credential_report()
        old_report_state = report_state
        report_state = cred_report_response["State"]
        if report_state != old_report_state:
            print(report_state, sep="")
        else:
            print(".", sep="")
        sys.stdout.flush()
        time.sleep(1)
    print()
    cred_report = get_credential_report()
    col_count = 3
    print(f"Got credentials report. Showing only the first {col_count} columns.")
    cred_lines = [
        line.split(",")[:col_count] for line in
cred_report.decode("utf-8").split("\n")
```

```
    ]
    col_width = max([len(item) for line in cred_lines for item in line]) + 2
    for line in cred_report.decode("utf-8").split("\n"):
        print(
            "".join(element.ljust(col_width) for element in line.split(",")
[:col_count])
        )

    print("-" * 88)
    print("Let's get an account summary.")
    summary = get_summary()
    print("Here's your summary:")
    pprint.pprint(summary)

    print("-" * 88)
    print("Let's get authorization details!")
    details = get_authorization_details([])
    see_details = input("These are pretty long, do you want to see them (y/n)? ")
    if see_details.lower() == "y":
        pprint.pprint(details)

    print("-" * 88)
    pw_policy_created = None
    see_pw_policy = input("Want to see the password policy for the account (y/n)?
 ")
    if see_pw_policy.lower() == "y":
        while True:
            if print_password_policy():
                break
            else:
                answer = input(
                    "Do you want to create a default password policy (y/n)? "
                )
                if answer.lower() == "y":
                    pw_policy_created = iam.create_account_password_policy()
                else:
                    break
    if pw_policy_created is not None:
        answer = input("Do you want to delete the password policy (y/n)? ")
        if answer.lower() == "y":
            pw_policy_created.delete()
            print("Password policy deleted.")

    print("The SAML providers for your account are:")
```

```
    list_saml_providers(10)

    print("-" * 88)
    print("Thanks for watching.")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - CreateAccountAlias
  - DeleteAccountAlias
  - GenerateCredentialReport
  - GetAccountAuthorizationDetails
  - GetAccountSummary
  - GetCredentialReport
  - ListAccountAliases

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将硬编码的密钥移动到 Secrets Manager

以下代码示例展示了如何：

- 创建 IAM 角色
- 在 Secrets Manager 中创建密钥
- 更新您的应用程序代码。
- 更新密钥
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "============================================================"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
```

```
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "========================================="
    echo "RESOURCES CREATED"
    echo "========================================="

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "========================================="
    echo "CLEANUP CONFIRMATION"
    echo "========================================="
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
```

```
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs
```

```
# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
```

```
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
```

```
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "======================================="
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi
```

```
echo ""
echo "============================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity

- GetSecretValue
- PutResourcePolicy
- UpdateSecret

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

允许 Amazon Compute Optimizer 自动化功能应用建议的操作

以下代码示例演示了如何……此基于权限的策略允许 Amazon Compute Optimizer 自动化功能应用建议的操作

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "aco-automation.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

在整个组织中启用自动化功能的策略

以下代码示例展示如何通过此基于权限的策略在整个组织中启用自动化功能

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:AssociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:DisassociateAccounts",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:ListAccounts",
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

启用账户自动化功能的策略

以下代码示例展示如何通过此基于权限的策略启用账户自动化功能

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:PutRolePolicy",
                "iam:AttachRolePolicy"
            ],
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        },
        {
            "Effect": "Allow",
            "Action": "aco-automation:UpdateEnrollmentConfiguration",
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

向组织管理账户授予对 Compute Optimizer 自动化功能完全访问权限的策略

以下代码示例展示如何通过此基于权限的策略向组织的管理账户授予对 Compute Optimizer 自动化功能的完全访问权限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
               "ec2:DescribeVolumes",
               "organizations:ListAccounts",
               "organizations:DescribeOrganization",
               "organizations:DescribeAccount",
               "organizations:EnableAWSServiceAccess",
               "organizations:ListDelegatedAdministrators",
               "organizations:RegisterDelegatedAdministrator",
               "organizations:DeregisterDelegatedAdministrator"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

向独立 Amazon 账户授予对 Compute Optimizer 自动化功能完全访问权限的策略

以下代码示例展示如何通过此基于权限的策略向独立 Amazon 账户授予对 Compute Optimizer 自动化功能的完全访问权限

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:*",
            "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

向组织管理账户授予对 Compute Optimizer 自动化功能只读访问权限的策略

以下代码示例展示如何通过此基于权限的策略向组织的管理账户授予对 Compute Optimizer 自动化功能的只读访问权限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
               "aco-automation:GetEnrollmentConfiguration",
               "aco-automation:GetAutomationEvent",
               "aco-automation:GetAutomationRule",
               "aco-automation:ListAccounts",
               "aco-automation:ListAutomationEvents",
               "aco-automation:ListAutomationEventSteps",
               "aco-automation:ListAutomationEventSummaries",
               "aco-automation:ListAutomationRules",
               "aco-automation:ListAutomationRulePreview",
               "aco-automation:ListAutomationRulePreviewSummaries",
               "aco-automation:ListRecommendedActions",
               "aco-automation:ListRecommendedActionSummaries",
               "aco-automation:ListTagsForResource",
```

```
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

向独立 Amazon 账户授予对 Compute Optimizer 自动化功能只读访问权限的策略

以下代码示例展示如何通过此基于权限的策略向独立 Amazon 账户授予对 Compute Optimizer 自动化功能的只读访问权限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "aco-automation:GetEnrollmentConfiguration",
                "aco-automation:GetAutomationEvent",
                "aco-automation:GetAutomationRule",
                "aco-automation:ListAutomationEvents",
                "aco-automation:ListAutomationEventSteps",
                "aco-automation:ListAutomationEventSummaries",
                "aco-automation:ListAutomationRules",
                "aco-automation:ListAutomationRulePreview",
                "aco-automation:ListAutomationRulePreviewSummaries",
                "aco-automation:ListRecommendedActions",
                "aco-automation:ListRecommendedActionSummaries",
                "aco-automation:ListTagsForResource",
                "ec2:DescribeVolumes"
            ],
            "Resource": "*"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

授予计算优化自动化功能的服务相关角色权限的策略

以下代码示例展示如何通过此基于权限的策略授予计算优化自动化功能的服务相关角色权限

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceLinkedRole",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation",
            "Condition": {"StringLike": {"iam:AWSServiceName": "aco-
automation.amazonaws.com"}}
        },
        {
            "Effect": "Allow",
            "Action": "iam:PutRolePolicy",
            "Resource": "arn:aws:iam::*:role/aws-service-role/aco-
automation.amazonaws.com/AWSServiceRoleForComputeOptimizerAutomation"
        }
    ]
}
```

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon SDK 回滚 IAM policy 版本

以下代码示例展示了如何：

- 获取按日期顺序排序的策略版本列表。
- 查找默认策略版本。
- 将以前的策略版本设为默认版本。
- 删除旧的默认版本。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
def rollback_policy_version(policy_arn):
    """
    Rolls back to the previous default policy, if it exists.

    1. Gets the list of policy versions in order by date.
    2. Finds the default.
    3. Makes the previous policy the default.
    4. Deletes the old default version.

    :param policy_arn: The ARN of the policy to roll back.
    :return: The default version of the policy after the rollback.
    """
    try:
        policy_versions = sorted(
            iam.Policy(policy_arn).versions.all(),
            key=operator.attrgetter("create_date"),
        )
        logger.info("Got %s versions for %s.", len(policy_versions), policy_arn)
    except ClientError:
        logger.exception("Couldn't get versions for %s.", policy_arn)
        raise

    default_version = None
    rollback_version = None
    try:
        while default_version is None:
            ver = policy_versions.pop()
            if ver.is_default_version:
                default_version = ver
        rollback_version = policy_versions.pop()
        rollback_version.set_as_default()
```

```
        logger.info("Set %s as the default version.",
rollback_version.version_id)
        default_version.delete()
        logger.info("Deleted original default version %s.",
default_version.version_id)
    except IndexError:
        if default_version is None:
            logger.warning("No default version found for %s.", policy_arn)
        elif rollback_version is None:
            logger.warning(
                "Default version %s found for %s, but no previous version exists,
so "
                "nothing to roll back to.",
                default_version.version_id,
                policy_arn,
            )
    except ClientError:
        logger.exception("Couldn't roll back version for %s.", policy_arn)
        raise
    else:
        return rollback_version
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python (Boto3) API Reference》中的以下主题。
  - DeletePolicyVersion
  - ListPolicyVersions
  - SetDefaultPolicyVersion

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 FIS 在 EC2 实例上运行 CPU 压力测试

以下代码示例展示了如何：

- 创建 IAM 角色
- 创建 CloudWatch 告警

- 创建实验模板
- 运行试验
- 验证结果
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "============================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
```

```
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
```

```
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")
```

```
echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
```

```
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")
```

```
# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
```

```
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"
```

```
# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
```

```
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
```

```
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
```

```
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
echo "==========================================="
echo "RESOURCES CREATED"
echo "==========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "==========================================="
```

```
# Prompt for cleanup
echo ""
echo "========================================="
echo "CLEANUP CONFIRMATION"
echo "========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
```

```
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
 experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
```

```
echo "Log file: $LOG_FILE"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances
  - StartExperiment
  - StopExperiment
  - TerminateInstances
  - Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Command Line Interface v2 为 DynamoDB 设置基于属性的访问控制

以下代码示例展示了如何为 DynamoDB 实现基于属性的访问权限控制（ABAC）。

- 创建用于实现 ABAC 的 IAM 策略。
- 为不同部门创建带有标签的表。
- 根据标签列出并筛选表。

Bash

Amazon CLI 及 Bash 脚本

创建用于实现 ABAC 的 IAM 策略。

```
# Step 1: Create a policy document for ABAC
cat > abac-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:Scan"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
```

```
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
      "Resource": "arn:aws:dynamodb:*:*:table/*",
      "Condition": {
        "StringEquals": {
          "aws:ResourceTag/Department": "${aws:PrincipalTag/Department}",
          "aws:ResourceTag/Environment": "Development"
        }
      }
    }
  ]
}
EOF

# Step 2: Create the IAM policy
aws iam create-policy \
    --policy-name DynamoDBDepartmentBasedAccess \
    --policy-document file://abac-policy.json
```

为不同部门创建带有标签的表。

```
# Create a DynamoDB table with tags for ABAC
aws dynamodb create-table \
    --table-name FinanceData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --tags \
        Key=Department,Value=Finance \
        Key=Environment,Value=Development

# Create another table with different tags
aws dynamodb create-table \
    --table-name MarketingData \
    --attribute-definitions \
        AttributeName=RecordID,AttributeType=S \
    --key-schema \
        AttributeName=RecordID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
```

```
    --tags \
        Key=Department,Value=Marketing \
        Key=Environment,Value=Production
```

根据标签列出并筛选表。

```
# List all DynamoDB tables
echo "Listing all tables:"
aws dynamodb list-tables

# Get ARNs for all tables
echo -e "\nGetting ARNs for all tables:"
TABLE_ARNS=$(aws dynamodb list-tables --query "TableNames[*]" --output text |
 xargs -I {} aws dynamodb describe-table --table-name {} --query "Table.TableArn"
 --output text)

# For each table ARN, list its tags
echo -e "\nListing tags for each table:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    echo -e "\nTags for table: $TABLE_NAME"
    aws dynamodb list-tags-of-resource --resource-arn $ARN
done

# Example: Find tables with a specific tag
echo -e "\nFinding tables with Environment=Production tag:"
for ARN in $TABLE_ARNS; do
    TABLE_NAME=$(echo $ARN | awk -F/ '{print $2}')
    TAGS=$(aws dynamodb list-tags-of-resource --resource-arn $ARN --query "Tags[?
Key=='Environment' && Value=='Production']" --output text)
    if [ ! -z "$TAGS" ]; then
        echo "Table with Production tag: $TABLE_NAME"
    fi
done
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - CreatePolicy
  - CreateTable
  - ListTables

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

设置 Systems Manager

以下代码示例展示了如何：

- 为 Systems Manager 创建 IAM 权限
- 为 Systems Manager 创建 IAM 角色
- 配置 Systems Manager
- 验证设置
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service

# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
```

```
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
    echo "========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
```

```
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done
```

```
    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "QuickSetupActions",
       "Effect": "Allow",
       "Action": [
         "ssm-quicksetup:*"
       ],
       "Resource": "*"
```

```
    },
    {
      "Sid": "SsmReadOnly",
      "Effect": "Allow",
      "Action": [
        "ssm:DescribeAutomationExecutions",
        "ssm:GetAutomationExecution",
        "ssm:ListAssociations",
        "ssm:DescribeAssociation",
        "ssm:ListDocuments",
        "ssm:ListResourceDataSync",
        "ssm:DescribePatchBaselines",
        "ssm:GetPatchBaseline",
        "ssm:DescribeMaintenanceWindows",
        "ssm:DescribeMaintenanceWindowTasks"
      ],
      "Resource": "*"
    },
    {
      "Sid": "SsmDocument",
      "Effect": "Allow",
      "Action": [
        "ssm:GetDocument",
        "ssm:DescribeDocument"
      ],
      "Resource": [
        "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
        "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
      ]
    },
    {
      "Sid": "SsmEnableExplorer",
      "Effect": "Allow",
      "Action": "ssm:StartAutomationExecution",
      "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
    },
    {
      "Sid": "SsmExplorerRds",
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsSummary",
        "ssm:CreateResourceDataSync",
        "ssm:UpdateResourceDataSync"
      ],
```

```
      "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
    },
    {
      "Sid": "OrgsReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListRoots",
        "organizations:ListParents",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAWSServiceAccessForOrganization"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgsAdministration",
      "Effect": "Allow",
      "Action": [
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:DeregisterDelegatedAdministrator"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "organizations:ServicePrincipal": [
            "ssm.amazonaws.com",
            "ssm-quicksetup.amazonaws.com",
            "member.org.stacksets.cloudformation.amazonaws.com",
            "resource-explorer-2.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "CfnReadOnly",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ListStacks",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackSets",
```

```
        "cloudformation:DescribeOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgCfnAccess",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ActivateOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CfnStackActions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStackEvents",
        "cloudformation:GetTemplate",
        "cloudformation:RollbackStack",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStack"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*"
      ]
    },
    {
      "Sid": "CfnStackSetActions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStackInstances",
        "cloudformation:CreateStackSet",
        "cloudformation:DeleteStackInstances",
        "cloudformation:DeleteStackSet",
        "cloudformation:DescribeStackInstance",
        "cloudformation:DetectStackSetDrift",
        "cloudformation:ListStackInstanceResourceDrifts",
        "cloudformation:DescribeStackSet",
```

```
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:ListStackInstances",
        "cloudformation:ListStackSetOperations",
        "cloudformation:ListStackSetOperationResults",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStackSet"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*",
        "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
      ]
    },
    {
      "Sid": "ValidationReadonlyActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IamRolesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:GetRole",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetRolePolicy",
        "iam:ListRolePolicies"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ]
    },
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": [
```

```
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ssm.amazonaws.com",
            "ssm-quicksetup.amazonaws.com",
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "IamRolesPoliciesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "ArnEquals": {
          "iam:PolicyARN": [
            "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
            "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
          ]
        }
      }
    },
    {
      "Sid": "CfnStackSetsSLR",
      "Effect": "Allow",
      "Action": [
        "iam:CreateServiceLinkedRole"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
        "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
        "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
        "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
        "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
      ]
    }
  ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
```

```
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
```

```
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
```

```
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"

echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
```

```
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================="
echo "CREATED RESOURCES"
echo "============================================="
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
```

```
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

在 CloudWatch 控制面板中使用属性变量监控多个 Lambda 函数

以下代码示例展示了如何：

- 创建用于监控的 Lambda 函数
- 创建 CloudWatch 控制面板
- 将属性变量添加到控制面板
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# CloudWatch Dashboard with Lambda Function Variable Script
# This script creates a CloudWatch dashboard with a property variable for Lambda
 function names

set -euo pipefail

# Security: Set restrictive umask
umask 0077

# Set up logging with secure permissions
LOG_FILE="cloudwatch-dashboard-script-v4.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
echo "Starting script execution at $(date)" >> "$LOG_FILE"

# Function to log commands and their output (with sensitive data sanitization)
log_cmd() {
    local cmd="$1"
    local sanitized_cmd="${cmd//--password*/--password [REDACTED]}"
    sanitized_cmd="${sanitized_cmd//--secret*/--secret [REDACTED]}"
    echo "$(date): Running command: $sanitized_cmd" >> "$LOG_FILE"
    eval "$cmd" 2>&1 | tee -a "$LOG_FILE"
    return ${PIPESTATUS[0]}
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"
```

```
    if [ $cmd_status -ne 0 ] || echo "$cmd_output" | grep -qi "error"; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        # Sanitize output before logging
        local sanitized_output="${cmd_output//arn:aws:iam::[0-9]*/
arn:aws:iam::ACCOUNT_ID}"
        echo "Command output: $sanitized_output" | tee -a "$LOG_FILE"
        cleanup_resources
        exit 1
    fi
}

# Trap errors and cleanup
trap 'cleanup_resources' EXIT ERR INT TERM

# Function to clean up resources
cleanup_resources() {
    local exit_code=$?

    echo "" | tee -a "$LOG_FILE"
    echo "========================================" | tee -a "$LOG_FILE"
    echo "CLEANUP PROCESS" | tee -a "$LOG_FILE"
    echo "========================================" | tee -a "$LOG_FILE"

    if [ -n "${DASHBOARD_NAME:-}" ]; then
        echo "Deleting CloudWatch dashboard: $DASHBOARD_NAME" | tee -a
 "$LOG_FILE"
        aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" 2>&1
 >> "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION1:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION1" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${LAMBDA_FUNCTION2:-}" ]; then
        echo "Deleting Lambda function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
        aws lambda delete-function --function-name "$LAMBDA_FUNCTION2" 2>&1 >>
 "$LOG_FILE" || true
    fi

    if [ -n "${ROLE_NAME:-}" ]; then
```

```
        echo "Detaching policy from role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole 2>&1 >>
 "$LOG_FILE" || true

        echo "Deleting IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
        aws iam delete-role --role-name "$ROLE_NAME" 2>&1 >> "$LOG_FILE" || true
    fi

    # Clean up temporary files securely
    shred -vfz -n 3 trust-policy.json lambda_function.py lambda_function.zip 2>/
dev/null || rm -f trust-policy.json lambda_function.py lambda_function.zip

    echo "Cleanup completed." | tee -a "$LOG_FILE"

    return $exit_code
}

# Validate AWS CLI is installed and authenticated
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed" | tee -a "$LOG_FILE"
    exit 1
fi

if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not properly authenticated" | tee -a "$LOG_FILE"
    exit 1
fi

# Get AWS region with validation
AWS_REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION" | tee -a
 "$LOG_FILE"
else
    echo "Using AWS region: $AWS_REGION" | tee -a "$LOG_FILE"
fi

# Validate region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    echo "ERROR: Invalid AWS region format: $AWS_REGION" | tee -a "$LOG_FILE"
    exit 1
fi
```

```
# Generate unique identifiers using secure random with validation
RANDOM_ID=$(openssl rand -hex 6)
if [ -z "$RANDOM_ID" ] || [ ${#RANDOM_ID} -ne 12 ]; then
    echo "ERROR: Failed to generate valid random identifier" | tee -a "$LOG_FILE"
    exit 1
fi

DASHBOARD_NAME="LambdaMetricsDashboard-${RANDOM_ID}"
LAMBDA_FUNCTION1="TestFunction1-${RANDOM_ID}"
LAMBDA_FUNCTION2="TestFunction2-${RANDOM_ID}"
ROLE_NAME="LambdaExecutionRole-${RANDOM_ID}"

# Validate resource names don't exceed AWS limits
if [ ${#DASHBOARD_NAME} -gt 128 ] || [ ${#LAMBDA_FUNCTION1} -gt 64 ] ||
 [ ${#ROLE_NAME} -gt 64 ]; then
    echo "ERROR: Generated resource names exceed AWS limits" | tee -a "$LOG_FILE"
    exit 1
fi

echo "Using random identifier: $RANDOM_ID" | tee -a "$LOG_FILE"
echo "Dashboard name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda function names: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "IAM role name: $ROLE_NAME" | tee -a "$LOG_FILE"

# Create IAM role for Lambda functions
echo "Creating IAM role for Lambda..." | tee -a "$LOG_FILE"
TRUST_POLICY='{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}'

echo "$TRUST_POLICY" > trust-policy.json
chmod 600 trust-policy.json
```

```
# Validate JSON before use
if ! python3 -m json.tool trust-policy.json > /dev/null 2>&1; then
    echo "ERROR: Invalid trust policy JSON" | tee -a "$LOG_FILE"
    exit 1
fi

ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name '$ROLE_NAME' --assume-
role-policy-document file://trust-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=cloudwatch-streams --output json")
check_error "$ROLE_OUTPUT" $? "Failed to create IAM role"

ROLE_ARN=$(echo "$ROLE_OUTPUT" | python3 -c "import sys, json;
 print(json.load(sys.stdin)['Role']['Arn'])" 2>/dev/null)
if [ -z "$ROLE_ARN" ]; then
    echo "ERROR: Failed to extract Role ARN" | tee -a "$LOG_FILE"
    exit 1
fi
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Attach Lambda basic execution policy to the role
echo "Attaching Lambda execution policy to role..." | tee -a "$LOG_FILE"
POLICY_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name '$ROLE_NAME' --
policy-arn arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole")
check_error "$POLICY_OUTPUT" $? "Failed to attach policy to role"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..." | tee -a "$LOG_FILE"
sleep 10

# Create simple Python Lambda function code with security validation
echo "Creating Lambda function code..." | tee -a "$LOG_FILE"
cat > lambda_function.py << 'LAMBDA_EOF'
def handler(event, context):
    print("Lambda function executed successfully")
    return {
        'statusCode': 200,
        'body': 'Success'
    }
LAMBDA_EOF

chmod 600 lambda_function.py

# Validate Python syntax
if ! python3 -m py_compile lambda_function.py 2>/dev/null; then
```

```
    echo "ERROR: Invalid Python syntax in Lambda function" | tee -a "$LOG_FILE"
    exit 1
fi

# Zip the Lambda function code
zip -j -q lambda_function.zip lambda_function.py
if [ ! -f lambda_function.zip ]; then
    echo "ERROR: Failed to create lambda_function.zip" | tee -a "$LOG_FILE"
    exit 1
fi
chmod 600 lambda_function.zip

# Validate zip file integrity
if ! unzip -t lambda_function.zip > /dev/null 2>&1; then
    echo "ERROR: Created zip file is corrupted" | tee -a "$LOG_FILE"
    exit 1
fi

# Create first Lambda function
echo "Creating first Lambda function: $LAMBDA_FUNCTION1..." | tee -a "$LOG_FILE"
LAMBDA1_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION1' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA1_OUTPUT" $? "Failed to create first Lambda function"

# Create second Lambda function
echo "Creating second Lambda function: $LAMBDA_FUNCTION2..." | tee -a "$LOG_FILE"
LAMBDA2_OUTPUT=$(log_cmd "aws lambda create-function --function-name
 '$LAMBDA_FUNCTION2' --runtime python3.11 --role '$ROLE_ARN' --handler
 lambda_function.handler --zip-file fileb://lambda_function.zip --timeout 30 --
memory-size 128 --tags project=doc-smith,tutorial=cloudwatch-streams")
check_error "$LAMBDA2_OUTPUT" $? "Failed to create second Lambda function"

# Invoke Lambda functions to generate some metrics
echo "Invoking Lambda functions to generate metrics..." | tee -a "$LOG_FILE"
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION1' --payload '{}' /
dev/null" || true
log_cmd "aws lambda invoke --function-name '$LAMBDA_FUNCTION2' --payload '{}' /
dev/null" || true

# Create CloudWatch dashboard with property variable
echo "Creating CloudWatch dashboard with property variable..." | tee -a
 "$LOG_FILE"
```

```
# Create dashboard body with proper escaping and validation
DASHBOARD_BODY=$(cat <<EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "$LAMBDA_FUNCTION1" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$AWS_REGION",
        "title": "Lambda Invocations",
        "period": 300,
        "stat": "Sum"
      }
    }
  ]
}
EOF
)

# Validate JSON before sending
if ! echo "$DASHBOARD_BODY" | python3 -m json.tool > /dev/null 2>&1; then
    echo "ERROR: Dashboard body is not valid JSON" | tee -a "$LOG_FILE"
    exit 1
fi

# First create a basic dashboard without variables
echo "Creating initial dashboard without variables..." | tee -a "$LOG_FILE"
DASHBOARD_OUTPUT=$(aws cloudwatch put-dashboard --dashboard-name
 "$DASHBOARD_NAME" --dashboard-body "$DASHBOARD_BODY" --output json 2>&1)
check_error "$DASHBOARD_OUTPUT" $? "Failed to create initial CloudWatch
 dashboard"

# Now let's try to add a property variable using the console instructions
echo "To complete the tutorial, please follow these steps in the CloudWatch
 console:" | tee -a "$LOG_FILE"
```

```
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/" | tee -a "$LOG_FILE"
echo "2. Navigate to Dashboards and select your dashboard: $DASHBOARD_NAME" | tee
 -a "$LOG_FILE"
echo "3. Choose Actions > Variables > Create a variable" | tee -a "$LOG_FILE"
echo "4. Choose Property variable" | tee -a "$LOG_FILE"
echo "5. For Property that the variable changes, choose FunctionName" | tee -a
 "$LOG_FILE"
echo "6. For Input type, choose Select menu (dropdown)" | tee -a "$LOG_FILE"
echo "7. Choose Use the results of a metric search" | tee -a "$LOG_FILE"
echo "8. Choose Pre-built queries > Lambda > Errors" | tee -a "$LOG_FILE"
echo "9. Choose By Function Name and then choose Search" | tee -a "$LOG_FILE"
echo "10. (Optional) Configure any secondary settings as desired" | tee -a
 "$LOG_FILE"
echo "11. Choose Add variable" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "The dashboard has been created and can be accessed at:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"

# Verify dashboard creation
echo "Verifying dashboard creation..." | tee -a "$LOG_FILE"
VERIFY_OUTPUT=$(aws cloudwatch get-dashboard --dashboard-name "$DASHBOARD_NAME"
 --output json 2>&1)
check_error "$VERIFY_OUTPUT" $? "Failed to verify dashboard creation"

echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "DASHBOARD CREATED SUCCESSFULLY" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
echo "Dashboard Name: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "Lambda Functions: $LAMBDA_FUNCTION1, $LAMBDA_FUNCTION2" | tee -a
 "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "You can view your dashboard in the CloudWatch console:" | tee -a
 "$LOG_FILE"
echo "https://console.aws.amazon.com/cloudwatch/home#dashboards:name=
$DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"

# Auto-confirm cleanup
echo "" | tee -a "$LOG_FILE"
echo "==========================================" | tee -a "$LOG_FILE"
```

```
echo "CLEANUP CONFIRMATION" | tee -a "$LOG_FILE"
echo "========================================" | tee -a "$LOG_FILE"
echo "The following resources were created:" | tee -a "$LOG_FILE"
echo "- CloudWatch Dashboard: $DASHBOARD_NAME" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION1" | tee -a "$LOG_FILE"
echo "- Lambda Function: $LAMBDA_FUNCTION2" | tee -a "$LOG_FILE"
echo "- IAM Role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "" | tee -a "$LOG_FILE"
echo "Auto-confirming cleanup of all created resources..." | tee -a "$LOG_FILE"

echo "Script completed successfully." | tee -a "$LOG_FILE"
exit 0
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole
  - DetachRolePolicy
  - GetDashboard
  - Invoke
  - PutDashboard

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Command Line Interface v2 处理 DynamoDB Streams 和生存时间

以下代码示例演示了如何管理 DynamoDB Streams 和生存时间功能。

- 创建一个启用了 Streams 的表。
- 描述 Streams。
- 创建用于处理 Streams 的 Lambda 函数。
- 在表上启用 TTL。
- 添加具有 TTL 属性的项目。

- 描述 TTL 设置。

Bash

Amazon CLI 及 Bash 脚本

创建一个启用了 Streams 的表。

```
# Create a table with DynamoDB Streams enabled
aws dynamodb create-table \
    --table-name StreamsDemo \
    --attribute-definitions \
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST \
    --stream-specification StreamEnabled=true,StreamViewType=NEW_AND_OLD_IMAGES
```

描述 Streams。

```
# Get information about the stream
aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.StreamSpecification"

# Get the stream ARN
STREAM_ARN=$(aws dynamodb describe-table \
    --table-name StreamsDemo \
    --query "Table.LatestStreamArn" \
    --output text)

echo "Stream ARN: $STREAM_ARN"

# Describe the stream
aws dynamodbstreams describe-stream \
    --stream-arn $STREAM_ARN
```

为 Streams 创建 Lambda 函数。

```
# Step 1: Create an IAM role for the Lambda function
```

```
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

aws iam create-role \
    --role-name DynamoDBStreamsLambdaRole \
    --assume-role-policy-document file://trust-policy.json

# Step 2: Attach permissions to the role
aws iam attach-role-policy \
    --role-name DynamoDBStreamsLambdaRole \
    --policy-arn arn:aws:iam::aws:policy/service-role/
AWSLambdaDynamoDBExecutionRole

# Step 3: Create a Lambda function (code would be in a separate file)
echo "Lambda function creation would be done separately with appropriate code"

# Step 4: Create an event source mapping
echo "Example command to create event source mapping:"
echo "aws lambda create-event-source-mapping \\"
echo "    --function-name ProcessDynamoDBRecords \\"
echo "    --event-source $STREAM_ARN \\"
echo "    --batch-size 100 \\"
echo "    --starting-position LATEST"
```

在表上启用 TTL。

```
# Create a table for TTL demonstration
aws dynamodb create-table \
    --table-name TTLDemo \
    --attribute-definitions \
```

```
        AttributeName=ID,AttributeType=S \
    --key-schema \
        AttributeName=ID,KeyType=HASH \
    --billing-mode PAY_PER_REQUEST

# Wait for table to become active
aws dynamodb wait table-exists --table-name TTLDemo

# Enable TTL on the table
aws dynamodb update-time-to-live \
    --table-name TTLDemo \
    --time-to-live-specification "Enabled=true, AttributeName=ExpirationTime"
```

添加具有 TTL 属性的项目。

```
# Calculate expiration time (current time + 1 day in seconds)
EXPIRATION_TIME=$(date -d "+1 day" +%s)

# Add an item with TTL attribute
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item1"},
        "Data": {"S": "This item will expire in 1 day"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME'"}
    }'

# Add an item that expires in 1 hour
EXPIRATION_TIME_HOUR=$(date -d "+1 hour" +%s)
aws dynamodb put-item \
    --table-name TTLDemo \
    --item '{
        "ID": {"S": "item2"},
        "Data": {"S": "This item will expire in 1 hour"},
        "ExpirationTime": {"N": "'$EXPIRATION_TIME_HOUR'"}
    }'
```

描述 TTL 设置。

```
# Describe TTL settings for a table
aws dynamodb describe-time-to-live \
```

```
    --table-name TTLDemo
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateRole
  - CreateTable
  - DescribeTable
  - DescribeTimeToLive
  - PutItem
  - UpdateTimeToLive

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

通过 Amazon SDK 使用 IAM Policy Builder API

以下代码示例展示了如何：

- 使用面向对象的 API 创建 IAM policy。
- 将 IAM Policy Builder API 与 IAM 服务结合使用。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

示例使用以下导入。

```
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
import software.amazon.awssdk.policybuilder.iam.IamConditionOperator;
import software.amazon.awssdk.policybuilder.iam.IamEffect;
import software.amazon.awssdk.policybuilder.iam.IamPolicy;
import software.amazon.awssdk.policybuilder.iam.IamPolicyWriter;
import software.amazon.awssdk.policybuilder.iam.IamPrincipal;
import software.amazon.awssdk.policybuilder.iam.IamPrincipalType;
import software.amazon.awssdk.policybuilder.iam.IamResource;
import software.amazon.awssdk.policybuilder.iam.IamStatement;
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.iam.IamClient;
import software.amazon.awssdk.services.iam.model.GetPolicyResponse;
import software.amazon.awssdk.services.iam.model.GetPolicyVersionResponse;
import software.amazon.awssdk.services.sts.StsClient;

import java.net.URLDecoder;
import java.nio.charset.StandardCharsets;
import java.util.Arrays;
import java.util.List;
```

创建基于时间的策略。

```
        public String timeBasedPolicyExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")
                                                .addResource(IamResource.ALL)
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_GREATER_THAN)

 .key("aws:CurrentTime")

 .value("2020-04-01T00:00:00Z"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.DATE_LESS_THAN)

 .key("aws:CurrentTime")

 .value("2020-06-30T23:59:59Z")))
                                .build();
```

```
                // Use an IamPolicyWriter to write out the JSON string to a more
 readable
                // format.
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true)
                                .build());
        }
```

创建具有多个条件的策略。

```
        public String multipleConditionsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:GetItem")

 .addAction("dynamodb:BatchGetItem")
                                                .addAction("dynamodb:Query")
                                                .addAction("dynamodb:PutItem")
                                                .addAction("dynamodb:UpdateItem")
                                                .addAction("dynamodb:DeleteItem")

 .addAction("dynamodb:BatchWriteItem")

 .addResource("arn:aws:dynamodb:*:*:table/table-name")

 .addConditions(IamConditionOperator.STRING_EQUALS

 .addPrefix("ForAllValues:"),

 "dynamodb:Attributes",
                                                                        List.of("column-
name1", "column-name2", "column-name3"))
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS

 .addSuffix("IfExists"))

 .key("dynamodb:Select")
```

```
 .value("SPECIFIC_ATTRIBUTES")))
                                .build();

                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

在策略中使用主体。

```
        public String specifyPrincipalsExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.DENY)
                                                .addAction("s3:*")
                                                .addPrincipal(IamPrincipal.ALL)
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.ARN_NOT_EQUALS)

 .key("aws:PrincipalArn")

 .value("arn:aws:iam::444455556666:user/user-name")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

允许跨账户存取。

```
        public String allowCrossAccountAccessExample() {
                IamPolicy policy = IamPolicy.builder()
                                .addStatement(b -> b
                                                .effect(IamEffect.ALLOW)

 .addPrincipal(IamPrincipalType.AWS, "111122223333")
                                                .addAction("s3:PutObject")
```

```
                                                .addResource("arn:aws:s3:::amzn-
s3-demo-bucket/*")
                                                .addCondition(b1 -> b1

 .operator(IamConditionOperator.STRING_EQUALS)
                                                                    .key("s3:x-amz-
acl")
                                                                    .value("bucket-
owner-full-control")))
                                .build();
                return policy.toJson(IamPolicyWriter.builder()
                                .prettyPrint(true).build());
        }
```

生成并上传 IamPolicy。

```
        public String createAndUploadPolicyExample(IamClient iam, String
 accountID, String policyName) {
                // Build the policy.
                IamPolicy policy = IamPolicy.builder() // 'version' defaults to
 "2012-10-17".
                                .addStatement(IamStatement.builder()
                                                .effect(IamEffect.ALLOW)
                                                .addAction("dynamodb:PutItem")

 .addResource("arn:aws:dynamodb:us-east-1:" + accountID
                                                                    + ":table/
exampleTableName")
                                                .build())
                                .build();
                // Upload the policy.
                iam.createPolicy(r ->
 r.policyName(policyName).policyDocument(policy.toJson()));
                return
 policy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

下载并使用 IamPolicy。

```
        public String createNewBasedOnExistingPolicyExample(IamClient iam, String
 accountID, String policyName,
```

```
                        String newPolicyName) {

                String policyArn = "arn:aws:iam::" + accountID + ":policy/" +
policyName;
                GetPolicyResponse getPolicyResponse = iam.getPolicy(r ->
r.policyArn(policyArn));

                String policyVersion =
getPolicyResponse.policy().defaultVersionId();
                GetPolicyVersionResponse getPolicyVersionResponse = iam
                                .getPolicyVersion(r ->
r.policyArn(policyArn).versionId(policyVersion));

                // Create an IamPolicy instance from the JSON string returned
from IAM.
                String decodedPolicy =
URLDecoder.decode(getPolicyVersionResponse.policyVersion().document(),
                                StandardCharsets.UTF_8);
                IamPolicy policy = IamPolicy.fromJson(decodedPolicy);

                /*
                 * All IamPolicy components are immutable, so use the copy method
that creates a
                 * new instance that
                 * can be altered in the same method call.
                 *
                 * Add the ability to get an item from DynamoDB as an additional
action.
                 */
                IamStatement newStatement = policy.statements().get(0).copy(s ->
s.addAction("dynamodb:GetItem"));

                // Create a new statement that replaces the original statement.
                IamPolicy newPolicy = policy.copy(p ->
p.statements(Arrays.asList(newStatement)));

                // Upload the new policy. IAM now has both policies.
                iam.createPolicy(r -> r.policyName(newPolicyName)
                                .policyDocument(newPolicy.toJson()));

                return
newPolicy.toJson(IamPolicyWriter.builder().prettyPrint(true).build());
        }
```

- 有关更多信息，请参阅《Amazon SDK for Java 2.x 开发人员指南》https://docs.amazonaws.cn/sdk-for-java/latest/developer-guide/feature-iam-policy-builder.html。
- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的以下主题。
  - CreatePolicy
  - GetPolicy
  - GetPolicyVersion

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包的 Amazon STS 代码示例

以下代码示例显示如何将 Amazon STS 与 Amazon 软件开发工具包（SDK）一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务 服务结合来完成特定任务的代码示例。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon STS

- 使用 Amazon SDK 的 Amazon STS 基础知识示例
  - 使用 Amazon 开发工具包的 Amazon STS 操作
    - 将 AssumeRole 与 Amazon SDK 或 CLI 配合使用
    - 将 AssumeRoleWithWebIdentity 与 CLI 配合使用
    - 将 DecodeAuthorizationMessage 与 CLI 配合使用
    - 将 GetFederationToken 与 CLI 配合使用
    - 将 GetSessionToken 与 Amazon SDK 或 CLI 配合使用
- 使用 Amazon 开发工具包的 Amazon STS 场景
  - 使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色

- 配置 Amazon ECS Service Connect
- 使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL
- 使用 Lambda 代理集成创建 REST API
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 EC2 启动类型的 Amazon ECS 服务
- 创建 Amazon Managed Grafana 工作区
- 使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌
- 开始使用 Amazon ECR
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon OpenSearch Service 入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- 终端用户消息发送推送服务入门
- IoT Core 入门
- WAF 入门
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 设置 Systems Manager

使用 Amazon SDK 的 Amazon STS 基础知识示例

以下代码示例展示了如何将 Amazon Security Token Service 的基本功能与 Amazon SDK 结合使用。

示例

- 使用 Amazon 开发工具包的 Amazon STS 操作
  - 将 AssumeRole 与 Amazon SDK 或 CLI 配合使用
  - 将 AssumeRoleWithWebIdentity 与 CLI 配合使用
  - 将 DecodeAuthorizationMessage 与 CLI 配合使用
  - 将 GetFederationToken 与 CLI 配合使用

- 将 GetSessionToken 与 Amazon SDK 或 CLI 配合使用

使用 Amazon 开发工具包的 Amazon STS 操作

以下代码示例演示了如何使用 Amazon 开发工具包来执行各个 Amazon STS 操作。每个示例都包含一个指向 GitHub 的链接，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon STS API，是必须在上下文中运行的大型程序的代码节选。您可以在使用 Amazon 开发工具包的 Amazon STS 场景中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Security Token Service API 参考。

示例

- 将 AssumeRole 与 Amazon SDK 或 CLI 配合使用
- 将 AssumeRoleWithWebIdentity 与 CLI 配合使用
- 将 DecodeAuthorizationMessage 与 CLI 配合使用
- 将 GetFederationToken 与 CLI 配合使用
- 将 GetSessionToken 与 Amazon SDK 或 CLI 配合使用

将 **`AssumeRole`** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssumeRole`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 代入需要 MFA 令牌的 IAM 角色
- 为联合用户构建 URL

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
using System;
using System.Threading.Tasks;
using Amazon;
using Amazon.SecurityToken;
using Amazon.SecurityToken.Model;

namespace AssumeRoleExample
{
    class AssumeRole
    {
        /// <summary>
        /// This example shows how to use the AWS Security Token
        /// Service (AWS STS) to assume an IAM role.
        ///
        /// NOTE: It is important that the role that will be assumed has a
        /// trust relationship with the account that will assume the role.
        ///
        /// Before you run the example, you need to create the role you want to
        /// assume and have it trust the IAM account that will assume that role.
        ///
        /// See https://docs.aws.amazon.com/IAM/latest/UserGuide/
id_roles_create.html
        /// for help in working with roles.
        /// </summary>

        // A region property may be used if the profile or credentials loaded do
 not specify a region,
        // or to use a specific region.
        private static readonly RegionEndpoint REGION = RegionEndpoint.USWest2;

        static async Task Main()
        {
            // Create the SecurityToken client and then display the identity of
 the
            // default user.
            var roleArnToAssume = "arn:aws:iam::123456789012:role/
testAssumeRole";

            var client = new
 Amazon.SecurityToken.AmazonSecurityTokenServiceClient(REGION);

            // Get and display the information about the identity of the default
 user.
```

```
            var callerIdRequest = new GetCallerIdentityRequest();
            var caller = await client.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"Original Caller: {caller.Arn}");

            // Create the request to use with the AssumeRoleAsync call.
            var assumeRoleReq = new AssumeRoleRequest()
            {
                DurationSeconds = 1600,
                RoleSessionName = "Session1",
                RoleArn = roleArnToAssume
            };

            var assumeRoleRes = await client.AssumeRoleAsync(assumeRoleReq);

            // Now create a new client based on the credentials of the caller
 assuming the role.
            var client2 = new AmazonSecurityTokenServiceClient(credentials:
 assumeRoleRes.Credentials, REGION);

            // Get and display information about the caller that has assumed the
 defined role.
            var caller2 = await client2.GetCallerIdentityAsync(callerIdRequest);
            Console.WriteLine($"AssumedRole Caller: {caller2.Arn}");
        }
    }
}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API Reference》中的 AssumeRole。

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
##############################################################################
# function iecho
#
# This function enables the script to display the specified text only if
# the global variable $VERBOSE is set to true.
##############################################################################
function iecho() {
  if [[ $VERBOSE == true ]]; then
    echo "$@"
  fi
}

##############################################################################
# function errecho
#
# This function outputs everything sent to it to STDERR (standard error output).
##############################################################################
function errecho() {
  printf "%s\n" "$*" 1>&2
}

##############################################################################
# function sts_assume_role
#
# This function assumes a role in the AWS account and returns the temporary
#  credentials.
#
# Parameters:
#       -n role_session_name -- The name of the session.
#       -r role_arn -- The ARN of the role to assume.
#
# Returns:
#       [access_key_id, secret_access_key, session_token]
#     And:
#       0 - If successful.
#       1 - If an error occurred.
##############################################################################
function sts_assume_role() {
  local role_session_name role_arn response
  local option OPTARG # Required to use getopts command in a function.

  # bashsupport disable=BP5008
  function usage() {
```

```
    echo "function sts_assume_role"
    echo "Assumes a role in the AWS account and returns the temporary
 credentials:"
    echo "  -n role_session_name -- The name of the session."
    echo "  -r role_arn -- The ARN of the role to assume."
    echo ""
  }

  while getopts n:r:h option; do
    case "${option}" in
      n) role_session_name=${OPTARG} ;;
      r) role_arn=${OPTARG} ;;
      h)
        usage
        return 0
        ;;
      \?)
        echo "Invalid parameter"
        usage
        return 1
        ;;
    esac
  done

  response=$(aws sts assume-role \
    --role-session-name "$role_session_name" \
    --role-arn "$role_arn" \
    --output text \
    --query "Credentials.[AccessKeyId, SecretAccessKey, SessionToken]")

  local error_code=${?}

  if [[ $error_code -ne 0 ]]; then
    aws_cli_error_log $error_code
    errecho "ERROR: AWS reports create-role operation failed.\n$response"
    return 1
  fi

  echo "$response"

  return 0
}
```

- 有关 API 详细信息，请参阅《Amazon CLI Command Reference》中的 AssumeRole。

C++

SDK for C++

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
bool AwsDoc::STS::assumeRole(const Aws::String &roleArn,
                             const Aws::String &roleSessionName,
                             const Aws::String &externalId,
                             Aws::Auth::AWSCredentials &credentials,
                             const Aws::Client::ClientConfiguration
 &clientConfig) {
    Aws::STS::STSClient sts(clientConfig);
    Aws::STS::Model::AssumeRoleRequest sts_req;

    sts_req.SetRoleArn(roleArn);
    sts_req.SetRoleSessionName(roleSessionName);
    sts_req.SetExternalId(externalId);

    const Aws::STS::Model::AssumeRoleOutcome outcome = sts.AssumeRole(sts_req);

    if (!outcome.IsSuccess()) {
        std::cerr << "Error assuming IAM role. " <<
                  outcome.GetError().GetMessage() << std::endl;
    }
    else {
        std::cout << "Credentials successfully retrieved." << std::endl;
        const Aws::STS::Model::AssumeRoleResult result = outcome.GetResult();
        const Aws::STS::Model::Credentials &temp_credentials =
 result.GetCredentials();

        // Store temporary credentials in return argument.
        // Note: The credentials object returned by assumeRole differs
        // from the AWSCredentials object used in most situations.
        credentials.SetAWSAccessKeyId(temp_credentials.GetAccessKeyId());
        credentials.SetAWSSecretKey(temp_credentials.GetSecretAccessKey());
```

```
        credentials.SetSessionToken(temp_credentials.GetSessionToken());
    }

    return outcome.IsSuccess();
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API Reference》中的 AssumeRole。

CLI

Amazon CLI

要代入角色

以下 assume-role 命令将检索 IAM 角色 s3-access-example 的一组短期凭证。

```
aws sts assume-role \
    --role-arn arn:aws:iam::123456789012:role/xaccounts3access \
    --role-session-name s3-access-example
```

输出：

```
{
    "AssumedRoleUser": {
        "AssumedRoleId": "AROA3XFRBF535PLBIFPI4:s3-access-example",
        "Arn": "arn:aws:sts::123456789012:assumed-role/xaccounts3access/s3-
access-example"
    },
    "Credentials": {
        "SecretAccessKey": "9drTJvcXLB89EXAMPLELB8923FB892xMFI",
        "SessionToken": "AQoXdzELDDY//////////
wEaoAK1wvxJY12r2IrDFT2IvAzTCn3zHoZ7YNtpiQLF0MqZye/
qwjzP2iEXAMPLEbw/m3hsj8VBTkPORGvr9jM5sgP+w9IZWZnU+LWhmg
+a5fDi2oTGUYcdg9uexQ4mtCHIHfi4citgqZTgco40Yqr4lIlo4V2b2Dyauk0eYFNebHtYlFVgAUj
+7Indz3LU0aTWk1WKIjHmmMCIoTkyYp/k7kUG7moeEYKSitwQIi6Gjn+nyzM
+PtoA3685ixzv0R7i5rjQi0YE0lf1oeie3bDiNHncmzosRM6SFiPzSvp6h/32xQuZsjcypmwsPSDtTPYcs0+YN/8B
IcrxSpnWEXAMPLEXSDFTAQAM6Dl9zR0tXoybnlrZIwMLlMi1Kcgo5OytwU=",
        "Expiration": "2016-03-15T00:05:07Z",
        "AccessKeyId": "ASIAJEXAMPLEXEG2JICEA"
    }
```

```
}
```

该命令的输出包含访问密钥、私有密钥和会话令牌，您可以使用它们对 Amazon 进行身份验证。

要使用 Amazon CLI，则可以设置与角色关联的命名配置文件。使用配置文件时，Amazon CLI 将调用 assume-role 并为您管理凭证。有关更多信息，请参阅《Amazon CLI 用户指南》中的在 Amazon CLI 中使用 IAM 角色。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AssumeRole。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.sts.StsClient;
import software.amazon.awssdk.services.sts.model.AssumeRoleRequest;
import software.amazon.awssdk.services.sts.model.StsException;
import software.amazon.awssdk.services.sts.model.AssumeRoleResponse;
import software.amazon.awssdk.services.sts.model.Credentials;
import java.time.Instant;
import java.time.ZoneId;
import java.time.format.DateTimeFormatter;
import java.time.format.FormatStyle;
import java.util.Locale;

/**
 * To make this code example work, create a Role that you want to assume.
 * Then define a Trust Relationship in the AWS Console. You can use this as an
 * example:
 *
 * {
 * "Version":"2012-10-17",
 * "Statement": [
 * {
```

```
 * "Effect": "Allow",
 * "Principal": {
 * "AWS": "<Specify the ARN of your IAM user you are using in this code example>"
 * },
 * "Action": "sts:AssumeRole"
 * }
 * ]
 * }
 *
 * For more information, see "Editing the Trust Relationship for an Existing
 * Role" in the AWS Directory Service guide.
 *
 * Also, set up your development environment, including your credentials.
 *
 * For information, see this documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class AssumeRole {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <roleArn> <roleSessionName>\s

                Where:
                    roleArn - The Amazon Resource Name (ARN) of the role to
 assume (for example, arn:aws:iam::000008047983:role/s3role).\s
                    roleSessionName - An identifier for the assumed role session
 (for example, mysession).\s
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String roleArn = args[0];
        String roleSessionName = args[1];
        Region region = Region.US_EAST_1;
        StsClient stsClient = StsClient.builder()
                .region(region)
                .build();
```

```
        assumeGivenRole(stsClient, roleArn, roleSessionName);
        stsClient.close();
    }

    public static void assumeGivenRole(StsClient stsClient, String roleArn,
 String roleSessionName) {
        try {
            AssumeRoleRequest roleRequest = AssumeRoleRequest.builder()
                    .roleArn(roleArn)
                    .roleSessionName(roleSessionName)
                    .build();

            AssumeRoleResponse roleResponse = stsClient.assumeRole(roleRequest);
            Credentials myCreds = roleResponse.credentials();

            // Display the time when the temp creds expire.
            Instant exTime = myCreds.expiration();
            String tokenInfo = myCreds.sessionToken();

            // Convert the Instant to readable date.
            DateTimeFormatter formatter =
 DateTimeFormatter.ofLocalizedDateTime(FormatStyle.SHORT)
                    .withLocale(Locale.US)
                    .withZone(ZoneId.systemDefault());

            formatter.format(exTime);
            System.out.println("The token " + tokenInfo + "  expires on " +
 exTime);

        } catch (StsException e) {
            System.err.println(e.getMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API Reference》中的 AssumeRole。

JavaScript

SDK for JavaScript (v3)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

创建客户端。

```
import { STSClient } from "@aws-sdk/client-sts";
// Set the AWS Region.
const REGION = "us-east-1";
// Create an AWS STS service client object.
export const client = new STSClient({ region: REGION });
```

代入 IAM 角色。

```
import { AssumeRoleCommand } from "@aws-sdk/client-sts";

import { client } from "../libs/client.js";

export const main = async () => {
  try {
    // Returns a set of temporary security credentials that you can use to
    // access Amazon Web Services resources that you might not normally
    // have access to.
    const command = new AssumeRoleCommand({
      // The Amazon Resource Name (ARN) of the role to assume.
      RoleArn: "ROLE_ARN",
      // An identifier for the assumed role session.
      RoleSessionName: "session1",
      // The duration, in seconds, of the role session. The value specified
      // can range from 900 seconds (15 minutes) up to the maximum session
      // duration set for the role.
      DurationSeconds: 900,
    });
    const response = await client.send(command);
    console.log(response);
```

```
  } catch (err) {
    console.error(err);
  }
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AssumeRole。

SDK for JavaScript (v2)

> **Note**
>
> 查看 GitHub，了解更多信息。查找完整示例，了解如何在 Amazon 代码示例存储库中进行设置和运行。

```
// Load the AWS SDK for Node.js
const AWS = require("aws-sdk");
// Set the region
AWS.config.update({ region: "REGION" });

var roleToAssume = {
  RoleArn: "arn:aws:iam::123456789012:role/RoleName",
  RoleSessionName: "session1",
  DurationSeconds: 900,
};
var roleCreds;

// Create the STS service object
var sts = new AWS.STS({ apiVersion: "2011-06-15" });

//Assume Role
sts.assumeRole(roleToAssume, function (err, data) {
  if (err) console.log(err, err.stack);
  else {
    roleCreds = {
      accessKeyId: data.Credentials.AccessKeyId,
      secretAccessKey: data.Credentials.SecretAccessKey,
      sessionToken: data.Credentials.SessionToken,
    };
    stsGetCallerIdentity(roleCreds);
  }
```

```
});

//Get Arn of current identity
function stsGetCallerIdentity(creds) {
  var stsParams = { credentials: creds };
  // Create STS service object
  var sts = new AWS.STS(stsParams);

  sts.getCallerIdentity({}, function (err, data) {
    if (err) {
      console.log(err, err.stack);
    } else {
      console.log(data.Arn);
    }
  });
}
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API Reference》中的 AssumeRole。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回一组临时凭证（访问密钥、秘密密钥和会话令牌），这些凭证可用于在一小时内访问请求用户通常可能无法访问的 Amazon 资源。返回的凭证具有所担任角色的访问策略和所提供策略允许的权限（您不能使用所提供策略授予超出所担任角色访问策略定义权限的权限）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

示例 2：返回一组有效期为一小时的临时凭证，其拥有的权限与所担任角色访问策略中定义的权限相同。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

示例 3：返回一组临时凭证，提供与用于执行 cmdlet 的用户凭证关联的 MFA 的序列号和生成的令牌。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

示例 4：返回一组临时凭证，其已承担客户账户中定义的角色。对于第三方可以担任的每个角色，客户账户必须使用每次担任该角色时都必须在 -ExternalId 参数中传递的标识符来创建角色。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AssumeRole。

Tools for PowerShell V5

示例 1：返回一组临时凭证（访问密钥、秘密密钥和会话令牌），这些凭证可用于在一小时内访问请求用户通常可能无法访问的 Amazon 资源。返回的凭证具有所担任角色的访问策略和所提供策略允许的权限（您不能使用所提供策略授予超出所担任角色访问策略定义权限的权限）。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -Policy "...JSON policy..." -DurationInSeconds 3600
```

示例 2：返回一组有效期为一小时的临时凭证，其拥有的权限与所担任角色访问策略中定义的权限相同。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600
```

示例 3：返回一组临时凭证，提供与用于执行 cmdlet 的用户凭证关联的 MFA 的序列号和生成的令牌。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -SerialNumber "GAHT12345678" -TokenCode "123456"
```

示例 4：返回一组临时凭证，其已承担客户账户中定义的角色。对于第三方可以担任的每个角色，客户账户必须使用每次担任该角色时都必须在 -ExternalId 参数中传递的标识符来创建角色。

```
Use-STSRole -RoleSessionName "Bob" -RoleArn "arn:aws:iam::123456789012:role/demo"
 -DurationInSeconds 3600 -ExternalId "ABC123"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 AssumeRole。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

代入需要 MFA 令牌的 IAM 角色并使用临时凭证列出该账户的 Amazon S3 存储桶。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
```

```
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API Reference》中的 AssumeRole。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
 # Creates an AWS Security Token Service (AWS STS) client with specified
credentials.
 # This is separated into a factory function so that it can be mocked for unit
testing.
 #
 # @param key_id [String] The ID of the access key used by the STS client.
```

```
 # @param key_secret [String] The secret part of the access key used by the STS
client.
 def create_sts_client(key_id, key_secret)
   Aws::STS::Client.new(access_key_id: key_id, secret_access_key: key_secret)
 end

 # Gets temporary credentials that can be used to assume a role.
 #
 # @param role_arn [String] The ARN of the role that is assumed when these
credentials
 #                          are used.
 # @param sts_client [AWS::STS::Client] An AWS STS client.
 # @return [Aws::AssumeRoleCredentials] The credentials that can be used to
assume the role.
 def assume_role(role_arn, sts_client)
   credentials = Aws::AssumeRoleCredentials.new(
     client: sts_client,
     role_arn: role_arn,
     role_session_name: 'create-use-assume-role-scenario'
   )
   @logger.info("Assumed role '#{role_arn}', got temporary credentials.")
   credentials
 end
```

- 有关 API 详细信息，请参阅《适用于 Ruby 的 Amazon SDK API Reference》中的 AssumeRole。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
async fn assume_role(config: &SdkConfig, role_name: String, session_name:
 Option<String>) {
    let provider = aws_config::sts::AssumeRoleProvider::builder(role_name)
```

```
        .session_name(session_name.unwrap_or("rust_sdk_example_session".into()))
        .configure(config)
        .build()
        .await;

    let local_config = aws_config::from_env()
        .credentials_provider(provider)
        .load()
        .await;
    let client = Client::new(&local_config);
    let req = client.get_caller_identity();
    let resp = req.send().await;
    match resp {
        Ok(e) => {
            println!("UserID :               {}",
 e.user_id().unwrap_or_default());
            println!("Account:               {}",
 e.account().unwrap_or_default());
            println!("Arn    :               {}", e.arn().unwrap_or_default());
        }
        Err(e) => println!("{:?}", e),
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Rust API Reference》中的 AssumeRole。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSSTS

    public func assumeRole(role: IAMClientTypes.Role, sessionName: String)
        async throws -> STSClientTypes.Credentials
    {
```

```
        let input = AssumeRoleInput(
            roleArn: role.arn,
            roleSessionName: sessionName
        )
        do {
            let output = try await stsClient.assumeRole(input: input)

            guard let credentials = output.credentials else {
                throw ServiceHandlerError.authError
            }

            return credentials
        } catch {
            print("Error assuming role: ", dump(error))
            throw error
        }
    }
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的 AssumeRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **AssumeRoleWithWebIdentity** 与 CLI 配合使用

以下代码示例演示如何使用 AssumeRoleWithWebIdentity。

CLI

Amazon CLI

获取通过 Web 身份（OAuth 2."0）进行身份验证的角色的短期凭证

以下 assume-role-with-web-identity 命令将检索 IAM 角色 app1 的一组短期凭证。使用由指定 Web 身份提供者提供的 Web 身份令牌对请求进行身份验证。两个附加策略应用于会话，以进一步限制用户可以执行的操作。返回的凭证在生成一个小时后过期。

```
aws sts assume-role-with-web-identity \
    --duration-seconds 3600 \
    --role-session-name "app1" \
    --provider-id "www.amazon.com" \
```

```
    --policy-arns "arn:aws:iam::123456789012:policy/
q=webidentitydemopolicy1","arn:aws:iam::123456789012:policy/
webidentitydemopolicy2" \
    --role-arn arn:aws:iam::123456789012:role/FederatedWebIdentityRole \
    --web-identity-token "Atza
%7CIQEBLjAsAhRFiXuWpUXuRvQ9PZL3GMFcYevydwIUFAHZwXZXXXXXXXXJnrulxKDHwy87oGKPznh0D6bEQZTSCz
CrKqjG7nPBjNIL016GGvuS5gSvPRUxWES3VYfm1wl7WTI7jn-Pcb6M-
buCgHhFOzTQxod27L9CqnOLio7N3gZAGpsp6n1-
AJBOCJckcyXe2c6uD0srOJeZlKUm2eTDVMf8IehDVI0r1QOnTV6KzzAI3OY87Vd_cVMQ"
```

输出：

```
{
    "SubjectFromWebIdentityToken": "amzn1.account.AF6RHO7KZU5XRVQJGXK6HB56KR2A",
    "Audience": "client.5498841531868486423.1548@apps.example.com",
    "AssumedRoleUser": {
        "Arn": "arn:aws:sts::123456789012:assumed-role/FederatedWebIdentityRole/
app1",
        "AssumedRoleId": "AROACLKWSDQRAOEXAMPLE:app1"
    },
    "Credentials": {
        "AccessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    },
    "Provider": "www.amazon.com"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的请求临时安全凭证。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 AssumeRoleWithWebIdentity。

PowerShell

适用于 PowerShell V4 的工具

示例 1：为已通过 Login with Amazon 身份提供者进行身份验证的用户返回一组临时凭证，有效期为一小时。这些凭证采用与角色 ARN 所标识角色关联的访问策略。或者，您可以将 JSON 策

略传递给 -Policy 参数，以进一步细化访问权限（您授予的权限不能超过与该角色关联的权限中可用的权限）。提供给 -WebIdentityToken 的值是身份提供者返回的唯一用户标识符。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 AssumeRoleWithWebIdentity。

Tools for PowerShell V5

示例 1：为已通过 Login with Amazon 身份提供者进行身份验证的用户返回一组临时凭证，有效期为一小时。这些凭证采用与角色 ARN 所标识角色关联的访问策略。或者，您可以将 JSON 策略传递给 -Policy 参数，以进一步细化访问权限（您授予的权限不能超过与该角色关联的权限中可用的权限）。提供给 -WebIdentityToken 的值是身份提供者返回的唯一用户标识符。

```
Use-STSWebIdentityRole -DurationInSeconds 3600 -ProviderId "www.amazon.com"
 -RoleSessionName "app1" -RoleArn "arn:aws:iam::123456789012:role/
FederatedWebIdentityRole" -WebIdentityToken "Atza...DVI0r1"
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 AssumeRoleWithWebIdentity。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **DecodeAuthorizationMessage** 与 CLI 配合使用

以下代码示例演示如何使用 DecodeAuthorizationMessage。

CLI

Amazon CLI

解码为响应请求而返回的编码授权消息

以下 decode-authorization-message 示例从为响应 Amazon Web Services 请求而返回的编码消息中解码有关请求授权状态的其他信息。

```
aws sts decode-authorization-message \
```

```
    --encoded-message EXAMPLEWodyRNrtlQARDip-
eTA6i6DrlUhHhPQrLWB_lAbl5pAKxl9mPDLexYcGBreyIKQC1BGBIpBKr3dFDkwqeO7e2NMk5j_hmzAiChJN-8oy3
Ojau7BMjOTWwOtHPhV_Zaz87yENdipr745EjQwRd5LaoL3vN8_5ZfA9UiBMKDgVh1gjqZJFUiQoubv78V1RbHNYnK
pOu3FZjwYStfvTb3GHs3-6rLribGO9jZOktkfE6vqxlFzLyeDr4P2ihC1wty9tArCvvGzIAUNmARQJ2VVWPxioqgo
JWP5pwe_mAyqh0NLw-r1S56YC_90onj9A80sNrHlI-
tIiNd7tgNTYzDuPQYD2FMDBnp82V9eVmYGtPp5NIeSpuf3fOHanFuBZgENxZQZ2dlH3xJGMTtYayzZrRXjiq_SfX9
FaoPIb8LmmKVBLpIB0iFhU9sEHPqKHVPi6jdxXqKaZaFGvYVmVOiuQdNQKuyk0p067POFrZECLjjOtNPBOZCcuEKE
```

输出：

```
{
    "DecodedMessage": "{\"allowed\":false,\"explicitDeny\":true,
\"matchedStatements\":{\"items\":[{\"statementId\":\"VisualEditor0\",\"effect
\":\"DENY\",\"principals\":{\"items\":[{\"value\":\"AROA123456789EXAMPLE
\"}]},\"principalGroups\":{\"items\":[]},\"actions\":{\"items\":[{\"value
\":\"ec2:RunInstances\"}]},\"resources\":{\"items\":[{\"value\":\"*
\"}]},\"conditions\":{\"items\":[]}}]},\"failures\":{\"items\":[]},
\"context\":{\"principal\":{\"id\":\"AROA123456789EXAMPLE:Ana\",\"arn
\":\"arn:aws:sts::111122223333:assumed-role/Developer/Ana\"},\"action\":
\"RunInstances\",\"resource\":\"arn:aws:ec2:us-east-1:111122223333:instance/*
\",\"conditions\":{\"items\":[{\"key\":\"ec2:MetadataHttpPutResponseHopLimit\",
\"values\":{\"items\":[{\"value\":\"2\"}]}},{\"key\":\"ec2:InstanceMarketType
\",\"values\":{\"items\":[{\"value\":\"on-demand\"}]}},{\"key\":\"aws:Resource
\",\"values\":{\"items\":[{\"value\":\"instance/*\"}]}},{\"key\":\"aws:Account
\",\"values\":{\"items\":[{\"value\":\"111122223333\"}]}},{\"key\":
\"ec2:AvailabilityZone\",\"values\":{\"items\":[{\"value\":\"us-east-1f\"}]}},
{\"key\":\"ec2:ebsOptimized\",\"values\":{\"items\":[{\"value\":\"false\"}]}},
{\"key\":\"ec2:IsLaunchTemplateResource\",\"values\":{\"items\":[{\"value\":
\"false\"}]}},{\"key\":\"ec2:InstanceType\",\"values\":{\"items\":[{\"value
\":\"t2.micro\"}]}},{\"key\":\"ec2:RootDeviceType\",\"values\":{\"items\":
[{\"value\":\"ebs\"}]}},{\"key\":\"aws:Region\",\"values\":{\"items\":[{\"value
\":\"us-east-1\"}]}},{\"key\":\"ec2:MetadataHttpEndpoint\",\"values\":{\"items
\":[{\"value\":\"enabled\"}]}},{\"key\":\"aws:Service\",\"values\":{\"items
\":[{\"value\":\"ec2\"}]}},{\"key\":\"ec2:InstanceID\",\"values\":{\"items\":
[{\"value\":\"*\"}]}},{\"key\":\"ec2:MetadataHttpTokens\",\"values\":{\"items
\":[{\"value\":\"required\"}]}},{\"key\":\"aws:Type\",\"values\":{\"items
\":[{\"value\":\"instance\"}]}},{\"key\":\"ec2:Tenancy\",\"values\":{\"items
\":[{\"value\":\"default\"}]}},{\"key\":\"ec2:Region\",\"values\":{\"items
\":[{\"value\":\"us-east-1\"}]}},{\"key\":\"aws:ARN\",\"values\":{\"items\":
[{\"value\":\"arn:aws:ec2:us-east-1:111122223333:instance/*\"}]}}]}}}"
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的策略评估逻辑。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 DecodeAuthorizationMessage。

PowerShell

适用于 PowerShell V4 的工具

示例 1：对为响应请求而返回的所提供编码消息内容中包含的其他信息进行解码。对其他信息进行编码的原因是，授权状态的详细信息可能构成请求操作的用户不应看到的特权信息。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 DecodeAuthorizationMessage。

Tools for PowerShell V5

示例 1：对为响应请求而返回的所提供编码消息内容中包含的其他信息进行解码。对其他信息进行编码的原因是，授权状态的详细信息可能构成请求操作的用户不应看到的特权信息。

```
Convert-STSAuthorizationMessage -EncodedMessage "...encoded message..."
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 DecodeAuthorizationMessage。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetFederationToken** 与 CLI 配合使用

以下代码示例演示如何使用 GetFederationToken。

CLI

Amazon CLI

使用 IAM 用户访问密钥凭证返回一组临时的安全凭证

以下 get-federation-token 示例返回用户的一组临时安全凭证（由访问密钥 ID、秘密访问密钥和安全令牌组成）。您必须使用 IAM 用户的长期安全凭证调用 GetFederationToken 操作。

```
aws sts get-federation-token \
    --name Bob \
    --policy file://myfile.json \
    --policy-arns arn=arn:aws:iam::aws:policy/AmazonS3ReadOnlyAccess \
    --duration-seconds 900
```

myfile.json 的内容：

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "elasticloadbalancing:Describe*",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:ListMetrics",
                "cloudwatch:GetMetricStatistics",
                "cloudwatch:Describe*"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": "autoscaling:Describe*",
            "Resource": "*"
        }
    ]
}
```

输出：

```
{
    "Credentials": {
```

```
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
        "SessionToken": "EXAMPLEpZ2luX2VjEGoaCXVzLXdlc3QtMiJIMEYCIQC/
W9pL5ArQyDD5JwFL3/h5+WGopQ24GEXweNctwhi9sgIhAMkg
+MZE35iWM8s4r5Lr25f9rSTVPFH98G42QQunWMTfKq0DCOP/////////
wEQAxoMNDUyOTI1MTcwNTA3Igxuy3AOpuuoLsk3MJwqgQPg8QOd9HuoClUxq26wnc/nm
+eZLjHDyGf2KUAHK2DuaS/nrGSEXAMPLE",
        "Expiration": "2023-12-20T02:06:07+00:00"
    },
    "FederatedUser": {
        "FederatedUserId": "111122223333:Bob",
        "Arn": "arn:aws:sts::111122223333:federated-user/Bob"
    },
    "PackedPolicySize": 36
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的请求临时安全凭证。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetFederationToken。

PowerShell

适用于 PowerShell V4 的工具

示例 1：使用“Bob”作为联合用户的名称，请求有效期为一小时的联合令牌。此名称可用于引用基于资源的策略（例如 Amazon S3 存储桶策略）中的联合用户名。以 JSON 格式提供的 IAM 策略可用于缩小 IAM 用户可用权限的范围。提供的策略授予的权限不能超过授予请求用户的权限，联合用户的最终权限是，基于传递的策略和 IAM 用户策略交集的限制性最强的集合。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetFederationToken。

Tools for PowerShell V5

示例 1：使用“Bob”作为联合用户的名称，请求有效期为一小时的联合令牌。此名称可用于引用基于资源的策略（例如 Amazon S3 存储桶策略）中的联合用户名。以 JSON 格式提供的 IAM 策略可用于缩小 IAM 用户可用权限的范围。提供的策略授予的权限不能超过授予请求用户的权限，联合用户的最终权限是，基于传递的策略和 IAM 用户策略交集的限制性最强的集合。

```
Get-STSFederationToken -Name "Bob" -Policy "...JSON policy..." -DurationInSeconds
 3600
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet 参考 (V5)》中的 GetFederationToken。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **GetSessionToken** 与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 GetSessionToken。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 获取需要 MFA 令牌的会话令牌

CLI

Amazon CLI

要获取 IAM 身份的一组短期凭证

以下 get-session-token 命令将检索进行调用的 IAM 身份的一组短期凭证。生成的凭证可用于策略要求多重身份验证（MFA）的请求。凭证在生成 15 分钟后过期。

```
aws sts get-session-token \
    --duration-seconds 900 \
    --serial-number "YourMFADeviceSerialNumber" \
    --token-code 123456
```

输出：

```
{
    "Credentials": {
        "AccessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "SecretAccessKey": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYzEXAMPLEKEY",
        "SessionToken": "AQoEXAMPLEH4aoAH0gNCAPyJxz4BlCFFxWNE1OPTgk5TthT
+FvwqnKwRcOIfrRh3c/LTo6UDdyJwOOvEVPvLXCrrrUtdnniCEXAMPLE/
```

```
IvU1dYUg2RVAJBanLiHb4IgRmpRV3zrkuWJOgQs8IZZaIv2BXIa2R4OlgkBN9bkUDNCJiBeb/
AXlzBBko7b15fjrBs2+cTQtpZ3CYWFXG8C5zqx37wnOE49mRl/+OtkIKGO7fAE",
        "Expiration": "2020-05-19T18:06:10+00:00"
    }
}
```

有关更多信息，请参阅《Amazon IAM 用户指南》中的请求临时安全凭证。

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的 GetSessionToken。

PowerShell

适用于 PowerShell V4 的工具

示例 1：返回包含在设定时间段内有效的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于请求临时凭证的凭证是根据当前 shell 默认值推断出来的。要指定其他凭证，请使用 -ProfileName 或 -AccessKey/-SecretKey 参数。

```
Get-STSSessionToken
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

示例 2：返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于发出请求的凭证是从指定的配置文件中获得的。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
```

```
EXAMPLEACCESSKEYID                          2/16/2015 9:12:28 PM
 examplesecretaccesskey...                  SamPleTokeN.....
```

示例 3：使用与其凭证在配置文件“myprofilename”中指定的账户关联的 MFA 设备的标识号和该设备提供的值，返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

输出：

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V4)》中的 GetSessionToken。

Tools for PowerShell V5

示例 1：返回包含在设定时间段内有效的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于请求临时凭证的凭证是根据当前 shell 默认值推断出来的。要指定其他凭证，请使用 -ProfileName 或 -AccessKey/-SecretKey 参数。

```
Get-STSSessionToken
```

输出：

```
AccessKeyId                             Expiration
 SecretAccessKey                        SessionToken
-----------                             ----------
 ---------------                        ------------
EXAMPLEACCESSKEYID                      2/16/2015 9:12:28 PM
 examplesecretaccesskey...              SamPleTokeN.....
```

示例 2：返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。用于发出请求的凭证是从指定的配置文件中获得的。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

示例 3：使用与其凭证在配置文件“myprofilename”中指定的账户关联的 MFA 设备的标识号和该设备提供的值，返回包含有效期为一小时的临时凭证的 **Amazon.RuntimeAWSCredentials** 实例。

```
Get-STSSessionToken -DurationInSeconds 3600 -ProfileName myprofile -SerialNumber
 YourMFADeviceSerialNumber -TokenCode 123456
```

输出：

```
AccessKeyId                              Expiration
 SecretAccessKey                         SessionToken
-----------                              ----------
 ---------------                         ------------
EXAMPLEACCESSKEYID                       2/16/2015 9:12:28 PM
 examplesecretaccesskey...               SamPleTokeN.....
```

- 有关 API 详细信息，请参阅《Amazon Tools for PowerShell Cmdlet Reference (V5)》中的 GetSessionToken。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

通过传递 MFA 令牌获取会话令牌，然后使用令牌列出账户的 Amazon S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的 GetSessionToken。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包的 Amazon STS 场景

以下代码示例显示如何通过 Amazon 开发工具包实施 Amazon STS 中的常见场景。这些场景向您展示了如何通过调用 Amazon STS 中的多个函数或与其他 Amazon Web Services 服务 结合来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色
- 配置 Amazon ECS Service Connect
- 使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL
- 使用 Lambda 代理集成创建 REST API
- 创建 Fargate 启动类型的 Amazon ECS Linux 任务
- 将函数名称作为变量创建 CloudWatch 控制面板
- 创建 EC2 启动类型的 Amazon ECS 服务
- 创建 Amazon Managed Grafana 工作区
- 使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌
- 开始使用 Amazon ECR
- 开始使用 Amazon EKS
- Amazon MSK 入门
- Amazon OpenSearch Service 入门
- Amazon SageMaker Feature Store 入门
- Config 入门
- 终端用户消息发送推送服务入门
- IoT Core 入门
- WAF 入门
- 将硬编码的密钥移动到 Secrets Manager
- 使用 FIS 在 EC2 实例上运行 CPU 压力测试
- 设置 Systems Manager

使用 Amazon 开发工具包代入需要具有 Amazon STS 的 MFA 令牌的 IAM 角色

以下代码示例显示了如何代入需要 MFA 令牌的角色。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建一个 IAM 角色，该角色授予列出 Amazon S3 存储桶的权限。
- 创建一个 IAM 用户，该用户仅在提供 MFA 凭证时才有权代入角色。
- 为该用户注册一个 MFA 设备。
- 代入该角色并使用临时凭证列出 S3 存储桶。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个 IAM 用户，注册一个 MFA 设备，然后创建一个角色，该角色授予列出 S3 存储桶的权限。用户仅具有代入该角色的权限。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual MFA device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates a role with a policy that lets the user assume the role and requires
 MFA.
    Creates a policy that allows listing Amazon S3 buckets.
```

```
    Attaches the policy to the role.
    Creates an inline policy for the user that lets the user assume the role.

    For demonstration purposes, the user is created in the same account as the
role,
    but in practice the user would likely be from another account.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
resource
                         that has permissions to create users, roles, and
policies
                         in the account.
    :return: The newly created user, user key, virtual MFA device, and role.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")

    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")
```

```
    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": user.arn},
                        "Action": "sts:AssumeRole",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(f"Created role {role.name} that requires MFA.")

    policy = iam_resource.create_policy(
        PolicyName=unique_name("policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn=policy.arn)
    print(f"Created policy {policy.policy_name} and attached it to the role.")
```

```
    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "sts:AssumeRole",
                        "Resource": role.arn,
                    }
                ],
            }
        ),
    )
    print(
        f"Created an inline policy for {user.name} that lets the user assume "
        f"the role."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device, role
```

显示不允许代入没有 MFA 令牌的角色。

```
def try_to_assume_role_without_mfa(assume_role_arn, session_name, sts_client):
    """
    Shows that attempting to assume the role without sending MFA credentials
 results
    in an AccessDenied error.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role to assume.
    :param session_name: The name of the STS session.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    print(f"Trying to assume the role without sending MFA credentials...")
```

```
    try:
        sts_client.assume_role(RoleArn=assume_role_arn,
 RoleSessionName=session_name)
        raise RuntimeError("Expected AccessDenied error.")
    except ClientError as error:
        if error.response["Error"]["Code"] == "AccessDenied":
            print("Got AccessDenied.")
        else:
            raise
```

代入授予列出 S3 存储桶权限的角色，传递所需的 MFA 令牌，并显示可以列出的存储桶。

```
def list_buckets_from_assumed_role_with_mfa(
    assume_role_arn, session_name, mfa_serial_number, mfa_totp, sts_client
):
    """
    Assumes a role from another account and uses the temporary credentials from
    that role to list the Amazon S3 buckets that are owned by the other account.
    Requires an MFA device serial number and token.

    The assumed role must grant permission to list the buckets in the other
 account.

    :param assume_role_arn: The Amazon Resource Name (ARN) of the role that
                            grants access to list the other account's buckets.
    :param session_name: The name of the STS session.
    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an ARN.
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn,
        RoleSessionName=session_name,
        SerialNumber=mfa_serial_number,
        TokenCode=mfa_totp,
    )
    temp_credentials = response["Credentials"]
```

```
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Listing buckets for the assumed role's account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

销毁为演示创建的资源。

```
def teardown(user, virtual_mfa_device, role):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        policy_name = attached.policy_name
        role.detach_policy(PolicyArn=attached.arn)
        attached.delete()
        print(f"Detached and deleted {policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device, role = setup(iam_resource)
    print(f"Created {user.name} and {role.name}.")
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
 aws_secret_access_key=user_key.secret
        )
        try_to_assume_role_without_mfa(role.arn, "demo-sts-session", sts_client)
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_from_assumed_role_with_mfa(
            role.arn,
            "demo-sts-session",
            virtual_mfa_device.serial_number,
            mfa_totp,
            sts_client,
        )
    finally:
        teardown(user, virtual_mfa_device, role)
        print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的 AssumeRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

配置 Amazon ECS Service Connect

以下代码示例展示了如何：

- 创建 VPC 基础设施
- 设置日志记录
- 创建 ECS 集群
- 配置 IAM 角色
- 使用 Service Connect 创建服务
- 验证部署
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# ECS Service Connect Tutorial Script v4 - Modified to use Default VPC
# This script creates an ECS cluster with Service Connect and deploys an nginx
 service
# Uses the default VPC to avoid VPC limits

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ECS Service Connect Tutorial"
LOG_FILE="ecs-service-connect-tutorial-v4-default-vpc.log"
REGION=${AWS_DEFAULT_REGION:-${AWS_REGION:-$(aws configure get region 2>/dev/
null)}}
```

```
if [ -z "$REGION" ]; then
    echo "ERROR: No AWS region configured."
    echo "Set one with: aws configure set region us-east-1"
    exit 1
fi
ENV_PREFIX="tutorial"
CLUSTER_NAME="${ENV_PREFIX}-cluster"
NAMESPACE_NAME="service-connect"

# Generate random suffix for unique resource names
RANDOM_SUFFIX=$(openssl rand -hex 6)

# Arrays to track created resources for cleanup
declare -a CREATED_RESOURCES=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    log "ERROR: Script failed at line $1"
    log "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set up error handling
trap 'handle_error $LINENO' ERR

# Function to add resource to tracking array
track_resource() {
    CREATED_RESOURCES+=("$1")
    log "Tracking resource: $1"
}

# Function to check if command output contains actual errors
check_for_errors() {
    local output="$1"
    local command_name="$2"

    # Check for specific AWS CLI error patterns, not just any occurrence of
 "error"
```

```
    if echo "$output" | grep -qi "An error occurred\|InvalidParameterException\|
AccessDenied\|ValidationException\|ResourceNotFoundException"; then
        log "ERROR in $command_name: $output"
        return 1
    fi
    return 0
}

# Function to get AWS account ID
get_account_id() {
    ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
    log "Using AWS Account ID: $ACCOUNT_ID"
}

# Function to wait for resources to be ready
wait_for_resource() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            log "Waiting for cluster $resource_id to be active..."
            local attempt=1
            local max_attempts=30
            while [ $attempt -le $max_attempts ]; do
                local status=$(aws ecs describe-clusters --clusters
 "$resource_id" --query 'clusters[0].status' --output text)
                if [ "$status" = "ACTIVE" ]; then
                    log "Cluster is now active"
                    return 0
                fi
                log "Cluster status: $status (attempt $attempt/$max_attempts)"
                sleep 10
                ((attempt++))
            done
            log "ERROR: Cluster did not become active within expected time"
            return 1
            ;;
        "service")
            log "Waiting for service $resource_id to be stable..."
            aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id"
            ;;
        "nat-gateway")
```

```
            log "Waiting for NAT Gateway $resource_id to be available..."
            aws ec2 wait nat-gateway-available --nat-gateway-ids "$resource_id"
            ;;
    esac
}

# Function to use default VPC infrastructure
setup_default_vpc_infrastructure() {
    log "Using default VPC infrastructure..."

    # Get default VPC
    VPC_ID=$(aws ec2 describe-vpcs --filters "Name=isDefault,Values=true" --query
 'Vpcs[0].VpcId' --output text)
    if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
        log "ERROR: No default VPC found. Please create a default VPC first."
        exit 1
    fi
    log "Using default VPC: $VPC_ID"

    # Get default subnets
    SUBNETS=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 "Name=default-for-az,Values=true" --query 'Subnets[].SubnetId' --output text)
    SUBNET_ARRAY=($SUBNETS)

    if [ ${#SUBNET_ARRAY[@]} -lt 2 ]; then
        log "ERROR: Need at least 2 subnets for ECS Service Connect. Found:
 ${#SUBNET_ARRAY[@]}"
        exit 1
    fi

    PUBLIC_SUBNET1=${SUBNET_ARRAY[0]}
    PUBLIC_SUBNET2=${SUBNET_ARRAY[1]}

    log "Using subnets: $PUBLIC_SUBNET1, $PUBLIC_SUBNET2"

    # Create security group for ECS tasks
    SG_OUTPUT=$(aws ec2 create-security-group \
        --group-name "${ENV_PREFIX}-ecs-sg-${RANDOM_SUFFIX}" \
        --description "Security group for ECS Service Connect tutorial" \
        --vpc-id "$VPC_ID" \
        --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-ecs-service-
connect}]' 2>&1)
    check_for_errors "$SG_OUTPUT" "create-security-group"
```

```
    SECURITY_GROUP_ID=$(echo "$SG_OUTPUT" | grep -o '"GroupId": "[^"]*"' | cut -
d'"' -f4)
    track_resource "SG:$SECURITY_GROUP_ID"
    log "Created security group: $SECURITY_GROUP_ID"

    # Add inbound rules to security group
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 443 \
        --cidr 0.0.0.0/0 >/dev/null 2>&1 || true

    log "Default VPC infrastructure setup completed"
}

# Function to create CloudWatch log groups
create_log_groups() {
    log "Creating CloudWatch log groups..."

    # Create log group for nginx container
    aws logs create-log-group --log-group-name "/ecs/service-connect-nginx"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-nginx created"
            track_resource "LOG_GROUP:/ecs/service-connect-nginx"
        else
            log "Log group /ecs/service-connect-nginx already exists"
        fi
    }

    # Create log group for service connect proxy
    aws logs create-log-group --log-group-name "/ecs/service-connect-proxy"
 --tags project=doc-smith,tutorial=amazon-ecs-service-connect 2>&1 | grep -v
 "ResourceAlreadyExistsException" || {
        if [ ${PIPESTATUS[0]} -eq 0 ]; then
            log "Log group /ecs/service-connect-proxy created"
            track_resource "LOG_GROUP:/ecs/service-connect-proxy"
```

```
        else
            log "Log group /ecs/service-connect-proxy already exists"
        fi
    }
}

# Function to create ECS cluster with Service Connect
create_ecs_cluster() {
    log "Creating ECS cluster with Service Connect..."

    CLUSTER_OUTPUT=$(aws ecs create-cluster \
        --cluster-name "$CLUSTER_NAME" \
        --service-connect-defaults namespace="$NAMESPACE_NAME" \
        --tags key=Environment,value=tutorial key=project,value=doc-smith
 key=tutorial,value=amazon-ecs-service-connect 2>&1)
    check_for_errors "$CLUSTER_OUTPUT" "create-cluster"

    track_resource "CLUSTER:$CLUSTER_NAME"
    log "Created ECS cluster: $CLUSTER_NAME"

    wait_for_resource "cluster" "$CLUSTER_NAME"

    # Track the Service Connect namespace that gets created
    # Wait a moment for the namespace to be created
    sleep 5
    NAMESPACE_ID=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_ID" && "$NAMESPACE_ID" != "None" ]]; then
        track_resource "NAMESPACE:$NAMESPACE_ID"
        log "Service Connect namespace created: $NAMESPACE_ID"
    fi
}

# Function to create IAM roles
create_iam_roles() {
    log "Creating IAM roles..."

    # Check if ecsTaskExecutionRole exists
    if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
        log "IAM role ecsTaskExecutionRole exists"
    else
```

```
        log "Creating ecsTaskExecutionRole..."
        aws iam create-role \
            --role-name ecsTaskExecutionRole \
            --assume-role-policy-document '{
                "Version":"2012-10-17",
                "Statement": [{
                    "Effect": "Allow",
                    "Principal": {"Service": "ecs-tasks.amazonaws.com"},
                    "Action": "sts:AssumeRole"
                }]
            }' >/dev/null 2>&1
        aws iam attach-role-policy \
            --role-name ecsTaskExecutionRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy >/dev/null 2>&1
        track_resource "ROLE:ecsTaskExecutionRole"
        aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskExecutionRole"
        sleep 10
    fi

    # Check if ecsTaskRole exists, create if not
    if aws iam get-role --role-name ecsTaskRole >/dev/null 2>&1; then
        log "IAM role ecsTaskRole exists"
    else
        log "IAM role ecsTaskRole does not exist, will create it"

        # Create trust policy for ECS tasks
        cat > /tmp/ecs-task-trust-policy.json << EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
        aws iam create-role \
            --role-name ecsTaskRole \
            --assume-role-policy-document file:///tmp/ecs-task-trust-policy.json
 >/dev/null

        track_resource "IAM_ROLE:ecsTaskRole"
        aws iam tag-role --role-name ecsTaskRole --tags Key=project,Value=doc-
smith Key=tutorial,Value=amazon-ecs-service-connect
        log "Created ecsTaskRole"

        # Wait for role to be available
        sleep 10
    fi
}

# Function to create task definition
create_task_definition() {
    log "Creating task definition..."

    # Create task definition JSON
    cat > /tmp/task-definition.json << EOF
{
    "family": "service-connect-nginx",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole",
    "taskRoleArn": "arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskRole",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "portMappings": [
                {
                    "containerPort": 80,
                    "protocol": "tcp",
                    "name": "nginx-port"
                }
            ],
            "essential": true,
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
```

```
                    "awslogs-group": "/ecs/service-connect-nginx",
                    "awslogs-region": "${REGION}",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

    TASK_DEF_OUTPUT=$(aws ecs register-task-definition --cli-input-json file:///
tmp/task-definition.json 2>&1)
    check_for_errors "$TASK_DEF_OUTPUT" "register-task-definition"

    TASK_DEF_ARN=$(echo "$TASK_DEF_OUTPUT" | grep -o '"taskDefinitionArn":
 "[^"]*"' | cut -d'"' -f4)
    track_resource "TASK_DEF:service-connect-nginx"
    log "Created task definition: $TASK_DEF_ARN"

    # Clean up temporary file
    rm -f /tmp/task-definition.json
}

# Function to create ECS service with Service Connect
create_ecs_service() {
    log "Creating ECS service with Service Connect..."

    # Create service definition JSON
    cat > /tmp/service-definition.json << EOF
{
    "serviceName": "service-connect-nginx-service",
    "cluster": "${CLUSTER_NAME}",
    "taskDefinition": "service-connect-nginx",
    "desiredCount": 1,
    "launchType": "FARGATE",
    "networkConfiguration": {
        "awsvpcConfiguration": {
            "subnets": ["${PUBLIC_SUBNET1}", "${PUBLIC_SUBNET2}"],
            "securityGroups": ["${SECURITY_GROUP_ID}"],
            "assignPublicIp": "ENABLED"
        }
    },
    "serviceConnectConfiguration": {
        "enabled": true,
```

```
        "namespace": "${NAMESPACE_NAME}",
        "services": [
            {
                "portName": "nginx-port",
                "discoveryName": "nginx",
                "clientAliases": [
                    {
                        "port": 80,
                        "dnsName": "nginx"
                    }
                ]
            }
        ],
        "logConfiguration": {
            "logDriver": "awslogs",
            "options": {
                "awslogs-group": "/ecs/service-connect-proxy",
                "awslogs-region": "${REGION}",
                "awslogs-stream-prefix": "ecs-service-connect"
            }
        }
    },
    "tags": [
        {
            "key": "Environment",
            "value": "tutorial"
        },
        {
            "key": "project",
            "value": "doc-smith"
        },
        {
            "key": "tutorial",
            "value": "amazon-ecs-service-connect"
        }
    ]
}
EOF

    SERVICE_OUTPUT=$(aws ecs create-service --cli-input-json file:///tmp/service-
definition.json 2>&1)
    check_for_errors "$SERVICE_OUTPUT" "create-service"

    track_resource "SERVICE:service-connect-nginx-service"
```

```
    log "Created ECS service: service-connect-nginx-service"

    wait_for_resource "service" "service-connect-nginx-service"

    # Clean up temporary file
    rm -f /tmp/service-definition.json
}

# Function to verify deployment
verify_deployment() {
    log "Verifying deployment..."

    # Check service status
    SERVICE_STATUS=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].status' --output text)
    log "Service status: $SERVICE_STATUS"

    # Check running tasks
    RUNNING_COUNT=$(aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].runningCount' --output text)
    log "Running tasks: $RUNNING_COUNT"

    # Get task ARN
    TASK_ARN=$(aws ecs list-tasks \
        --cluster "$CLUSTER_NAME" \
        --service-name "service-connect-nginx-service" \
        --query 'taskArns[0]' --output text)

    if [[ "$TASK_ARN" != "None" && -n "$TASK_ARN" ]]; then
        log "Task ARN: $TASK_ARN"

        # Try to get task IP address
        TASK_IP=$(aws ecs describe-tasks \
            --cluster "$CLUSTER_NAME" \
            --tasks "$TASK_ARN" \
            --query 'tasks[0].attachments[0].details[?
name==`privateIPv4Address`].value' \
            --output text 2>/dev/null || echo "")

        if [[ -n "$TASK_IP" && "$TASK_IP" != "None" ]]; then
```

```
            log "Task IP address: $TASK_IP"
        else
            log "Could not retrieve task IP address"
        fi
    fi

    # Check Service Connect namespace
    NAMESPACE_STATUS=$(aws servicediscovery list-namespaces \
        --filters Name=TYPE,Values=HTTP \
        --query "Namespaces[?Name=='$NAMESPACE_NAME'].Id" --output text 2>/dev/
null || echo "")

    if [[ -n "$NAMESPACE_STATUS" && "$NAMESPACE_STATUS" != "None" ]]; then
        log "Service Connect namespace '$NAMESPACE_NAME' is active"
    else
        log "Service Connect namespace '$NAMESPACE_NAME' not found or not active"
    fi

    # Display Service Connect configuration
    log "Service Connect configuration:"
    aws ecs describe-services \
        --cluster "$CLUSTER_NAME" \
        --services "service-connect-nginx-service" \
        --query 'services[0].serviceConnectConfiguration' 2>/dev/null || true
}

# Function to display created resources
display_resources() {
    echo ""
    echo "============================================="
    echo "CREATED RESOURCES"
    echo "============================================="
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "- $resource"
    done
    echo "============================================="
    echo ""
}

# Function to cleanup resources
cleanup_resources() {
    log "Starting cleanup process..."

    # Delete resources in reverse order of creation
```

```
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[i]}"
        resource_type=$(echo "$resource" | cut -d':' -f1)
        resource_id=$(echo "$resource" | cut -d':' -f2)

        log "Cleaning up $resource_type: $resource_id"

        case "$resource_type" in
            "SERVICE")
                aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --desired-count 0 2>&1 | grep -qi "error" && log "Warning: Failed
 to scale down service $resource_id"
                aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$resource_id" 2>/dev/null || true
                aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$resource_id" --force 2>&1 | grep -qi "error" && log "Warning: Failed to delete
 service $resource_id"
                ;;
            "TASK_DEF")
                TASK_DEF_ARNS=$(aws ecs list-task-definitions --family-prefix
 "$resource_id" --query 'taskDefinitionArns' --output text 2>/dev/null)
                for arn in $TASK_DEF_ARNS; do
                    aws ecs deregister-task-definition --task-definition "$arn"
 >/dev/null 2>&1 || true
                done
                ;;
            "ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "IAM_ROLE")
                aws iam detach-role-policy --role-name "$resource_id" --policy-
arn "arn:aws:iam::aws:policy/service-role/AmazonECSTaskExecutionRolePolicy" 2>/
dev/null || true
                aws iam delete-role --role-name "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete role $resource_id"
                ;;
            "CLUSTER")
                aws ecs delete-cluster --cluster "$resource_id" 2>&1 | grep -qi
 "error" && log "Warning: Failed to delete cluster $resource_id"
                ;;
```

```
            "SG")
                for attempt in 1 2 3 4 5; do
                    if aws ec2 delete-security-group --group-id "$resource_id"
 2>/dev/null; then
                        break
                    fi
                    log "Security group $resource_id still has dependencies,
 retrying in 30s ($attempt/5)..."
                    sleep 30
                done
                ;;
            "LOG_GROUP")
                aws logs delete-log-group --log-group-name "$resource_id" 2>&1 |
 grep -qi "error" && log "Warning: Failed to delete log group $resource_id"
                ;;
            "NAMESPACE")
                # First, delete any services in the namespace
                NAMESPACE_SERVICES=$(aws servicediscovery list-services \
                    --filters Name=NAMESPACE_ID,Values="$resource_id" \
                    --query 'Services[].Id' --output text 2>/dev/null || echo "")

                if [[ -n "$NAMESPACE_SERVICES" && "$NAMESPACE_SERVICES" !=
 "None" ]]; then
                    for service_id in $NAMESPACE_SERVICES; do
                        aws servicediscovery delete-service --id "$service_id" >/
dev/null 2>&1 || true
                        sleep 2
                    done
                fi

                # Then delete the namespace
                aws servicediscovery delete-namespace --id "$resource_id" >/dev/
null 2>&1 || true
                ;;
        esac

        sleep 2  # Brief pause between deletions
    done

    # Clean up temporary files
    rm -f /tmp/ecs-task-trust-policy.json
    rm -f /tmp/task-definition.json
    rm -f /tmp/service-definition.json
```

```
    log "Cleanup completed"
}

# Main execution
main() {
    log "Starting $SCRIPT_NAME v4 (Default VPC)"
    log "Region: $REGION"
    log "Log file: $LOG_FILE"

    # Get AWS account ID
    get_account_id

    # Setup infrastructure using default VPC
    setup_default_vpc_infrastructure

    # Create CloudWatch log groups
    create_log_groups

    # Create ECS cluster
    create_ecs_cluster

    # Create IAM roles
    create_iam_roles

    # Create task definition
    create_task_definition

    # Create ECS service
    create_ecs_service

    # Verify deployment
    verify_deployment

    log "Tutorial completed successfully!"

    # Display created resources
    display_resources

    # Ask user if they want to clean up
    echo ""
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    echo "Do you want to clean up all created resources? (y/n): "
```

```
    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        cleanup_resources
        log "All resources have been cleaned up"
    else
        log "Resources left intact. You can clean them up later by running the
 cleanup function."
        echo ""
        echo "To clean up resources later, you can use the AWS CLI commands or
 the AWS Management Console."
        echo "Remember to delete resources in the correct order to avoid
 dependency issues."
    fi
}

# Make script executable and run
chmod +x "$0"
main "$@"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateLogGroup
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteLogGroup
  - DeleteNamespace
  - DeleteRole
  - DeleteSecurityGroup
  - DeleteService
  - DeregisterTaskDefinition
  - DescribeClusters

- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- DetachRolePolicy
- GetCallerIdentity
- GetRole
- ListNamespaces
- ListServices
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包为联合用户构建具有 Amazon STS 的 URL

以下代码示例展示了如何：

- 创建一个 IAM 角色，该角色授予对当前账户的 Amazon S3 资源的只读访问权限。
- 从 Amazon 联合身份验证端点获取安全令牌。
- 构建一个可以用于通过联合凭证访问控制台的 URL。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个角色，该角色授予对当前账户的 S3 资源的只读访问权限。

```
def setup(iam_resource):
    """
    Creates a role that can be assumed by the current user.
    Attaches a policy that allows only Amazon S3 read-only access.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 instance
                         that has the permission to create a role.
    :return: The newly created role.
    """
    role = iam_resource.create_role(
        RoleName=unique_name("role"),
        AssumeRolePolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Principal": {"AWS": iam_resource.CurrentUser().arn},
                        "Action": "sts:AssumeRole",
                    }
                ],
            }
        ),
    )
    role.attach_policy(PolicyArn="arn:aws:iam::aws:policy/
AmazonS3ReadOnlyAccess")
    print(f"Created role {role.name}.")

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return role
```

从 Amazon 联合身份验证端点获取安全令牌并构建一个可用于通过联合凭证访问控制台的 URL。

```
def construct_federated_url(assume_role_arn, session_name, issuer, sts_client):
    """
    Constructs a URL that gives federated users direct access to the AWS
 Management
    Console.

    1. Acquires temporary credentials from AWS Security Token Service (AWS STS)
 that
       can be used to assume a role with limited permissions.
    2. Uses the temporary credentials to request a sign-in token from the
       AWS federation endpoint.
    3. Builds a URL that can be used in a browser to navigate to the AWS
 federation
       endpoint, includes the sign-in token for authentication, and redirects to
       the AWS Management Console with permissions defined by the role that was
       specified in step 1.

    :param assume_role_arn: The role that specifies the permissions that are
 granted.
                            The current user must have permission to assume the
 role.
    :param session_name: The name for the STS session.
    :param issuer: The organization that issues the URL.
    :param sts_client: A Boto3 STS instance that can assume the role.
    :return: The federated URL.
    """
    response = sts_client.assume_role(
        RoleArn=assume_role_arn, RoleSessionName=session_name
    )
    temp_credentials = response["Credentials"]
    print(f"Assumed role {assume_role_arn} and got temporary credentials.")

    session_data = {
        "sessionId": temp_credentials["AccessKeyId"],
        "sessionKey": temp_credentials["SecretAccessKey"],
        "sessionToken": temp_credentials["SessionToken"],
    }
    aws_federated_signin_endpoint = "https://signin.aws.amazon.com/federation"

    # Make a request to the AWS federation endpoint to get a sign-in token.
    # The requests.get function URL-encodes the parameters and builds the query
 string
    # before making the request.
```

```
    response = requests.get(
        aws_federated_signin_endpoint,
        params={
            "Action": "getSigninToken",
            "SessionDuration": str(datetime.timedelta(hours=12).seconds),
            "Session": json.dumps(session_data),
        },
    )
    signin_token = json.loads(response.text)
    print(f"Got a sign-in token from the AWS sign-in federation endpoint.")

    # Make a federated URL that can be used to sign into the AWS Management
 Console.
    query_string = urllib.parse.urlencode(
        {
            "Action": "login",
            "Issuer": issuer,
            "Destination": "https://console.aws.amazon.com/",
            "SigninToken": signin_token["SigninToken"],
        }
    )
    federated_url = f"{aws_federated_signin_endpoint}?{query_string}"
    return federated_url
```

销毁为演示创建的资源。

```
def teardown(role):
    """
    Removes all resources created during setup.

    :param role: The demo role.
    """
    for attached in role.attached_policies.all():
        role.detach_policy(PolicyArn=attached.arn)
        print(f"Detached {attached.policy_name}.")
    role.delete()
    print(f"Deleted {role.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(f"Welcome to the AWS Security Token Service federated URL demo.")
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    role = setup(iam_resource)
    sts_client = boto3.client("sts")
    try:
        federated_url = construct_federated_url(
            role.arn, "AssumeRoleDemoSession", "example.org", sts_client
        )
        print(
            "Constructed a federated URL that can be used to connect to the "
            "AWS Management Console with role-defined permissions:"
        )
        print("-" * 88)
        print(federated_url)
        print("-" * 88)
        _ = input(
            "Copy and paste the above URL into a browser to open the AWS "
            "Management Console with limited permissions. When done, press "
            "Enter to clean up and complete this demo."
        )
    finally:
        teardown(role)
        print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的 AssumeRole。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Lambda 代理集成创建 REST API

以下代码示例展示了如何：

- 为 Lambda 执行创建 IAM 角色
- 创建并部署 Lambda 函数
- 创建 REST API
- 配置 Lambda 代理集成
- 部署并测试 API
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

set -euo pipefail

# Simple API Gateway Lambda Integration Script
# This script creates a REST API with Lambda proxy integration

# Generate random identifiers
FUNCTION_NAME="GetStartedLambdaProxyIntegration-$(openssl rand -hex 4)"
ROLE_NAME="GetStartedLambdaBasicExecutionRole-$(openssl rand -hex 4)"
API_NAME="LambdaProxyAPI-$(openssl rand -hex 4)"

# Get AWS account info
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
REGION=$(aws configure get region || echo "us-east-1")

# Validate inputs
if [[ -z "$ACCOUNT_ID" ]] || [[ -z "$REGION" ]]; then
    echo "Error: Failed to retrieve AWS account information" >&2
    exit 1
fi
```

```
echo "Creating Lambda function code..."

# Create Lambda function code with input validation
cat > lambda_function.py << 'EOF'
import json
import logging

logger = logging.getLogger()
logger.setLevel(logging.INFO)

def lambda_handler(event, context):
    try:
        logger.info("Received event: %s", json.dumps(event))

        greeter = 'World'

        # Safely retrieve greeter from query string parameters
        query_params = event.get('queryStringParameters') or {}
        if isinstance(query_params, dict) and 'greeter' in query_params:
            greeter_value = query_params.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from multi-value headers
        multi_headers = event.get('multiValueHeaders') or {}
        if isinstance(multi_headers, dict) and 'greeter' in multi_headers:
            greeter_list = multi_headers.get('greeter', [])
            if isinstance(greeter_list, list) and greeter_list:
                greeter = " and ".join(str(g) for g in greeter_list if g)

        # Safely retrieve greeter from headers
        headers = event.get('headers') or {}
        if isinstance(headers, dict) and 'greeter' in headers:
            greeter_value = headers.get('greeter')
            if isinstance(greeter_value, str) and greeter_value:
                greeter = greeter_value

        # Safely retrieve greeter from body
        body = event.get('body')
        if body and isinstance(body, str):
            try:
                body_dict = json.loads(body)
                if isinstance(body_dict, dict) and 'greeter' in body_dict:
                    greeter_value = body_dict.get('greeter')
```

```
                    if isinstance(greeter_value, str) and greeter_value:
                        greeter = greeter_value
            except (json.JSONDecodeError, ValueError) as e:
                logger.warning("Failed to parse body: %s", str(e))

        # Sanitize greeter to prevent injection
        greeter = greeter.replace('"', '\\"').replace("'", "\\'")

        response = {
            "statusCode": 200,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"message": f"Hello, {greeter}!"})
        }

        logger.info("Response: %s", json.dumps(response))
        return response

    except Exception as e:
        logger.error("Unexpected error: %s", str(e), exc_info=True)
        return {
            "statusCode": 500,
            "headers": {
                "Content-Type": "application/json"
            },
            "body": json.dumps({"error": "Internal server error"})
        }
EOF

# Create deployment package
zip -q function.zip lambda_function.py || {
    echo "Error: Failed to create function.zip" >&2
    exit 1
}

echo "Creating IAM role..."

# Create IAM trust policy
cat > trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
```

```
      "Effect": "Allow",
      "Principal": {
        "Service": "lambda.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role with error handling
aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document file://trust-policy.json \
    --description "Temporary role for Lambda execution" || {
    echo "Error: Failed to create IAM role" >&2
    exit 1
}

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=apigateway-lambda-integration

# Attach execution policy
aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || {
    echo "Error: Failed to attach IAM policy" >&2
    exit 1
}

# Wait for role propagation
sleep 15

echo "Creating Lambda function..."

# Create Lambda function with Python 3.11 (more recent runtime)
aws lambda create-function \
    --function-name "$FUNCTION_NAME" \
    --runtime python3.11 \
    --role "arn:aws:iam::$ACCOUNT_ID:role/$ROLE_NAME" \
    --handler lambda_function.lambda_handler \
    --zip-file fileb://function.zip \
    --timeout 30 \
```

```
    --memory-size 128 \
    --environment "Variables={LOG_LEVEL=INFO}" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration || {
    echo "Error: Failed to create Lambda function" >&2
    exit 1
}

echo "Creating API Gateway..."

# Create REST API with minimum logging
API_RESPONSE=$(aws apigateway create-rest-api \
    --name "$API_NAME" \
    --endpoint-configuration types=REGIONAL \
    --description "API for Lambda proxy integration tutorial" \
    --tags project=doc-smith,tutorial=apigateway-lambda-integration \
    --output json)

API_ID=$(echo "$API_RESPONSE" | grep -o '"id": "[^"]*"' | head -1 | cut -d'"' -
f4)

if [[ -z "$API_ID" ]]; then
    echo "Error: Failed to create API Gateway" >&2
    exit 1
fi

# Get root resource ID
ROOT_RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 'items[?path==`/`].id' --output text)

# Create helloworld resource
aws apigateway create-resource \
    --rest-api-id "$API_ID" \
    --parent-id "$ROOT_RESOURCE_ID" \
    --path-part helloworld || {
    echo "Error: Failed to create resource" >&2
    exit 1
}

# Get resource ID
RESOURCE_ID=$(aws apigateway get-resources --rest-api-id "$API_ID" --query
 "items[?pathPart=='helloworld'].id" --output text)

# Create ANY method with no authorization (intentional for tutorial)
aws apigateway put-method \
```

```
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --authorization-type NONE || {
    echo "Error: Failed to create method" >&2
    exit 1
}

# Set up Lambda proxy integration
LAMBDA_URI="arn:aws:apigateway:$REGION:lambda:path/2015-03-31/functions/
arn:aws:lambda:$REGION:$ACCOUNT_ID:function:$FUNCTION_NAME/invocations"

aws apigateway put-integration \
    --rest-api-id "$API_ID" \
    --resource-id "$RESOURCE_ID" \
    --http-method ANY \
    --type AWS_PROXY \
    --integration-http-method POST \
    --uri "$LAMBDA_URI" || {
    echo "Error: Failed to create integration" >&2
    exit 1
}

# Grant API Gateway permission to invoke Lambda
STATEMENT_ID="apigateway-invoke-$(openssl rand -hex 4)"
SOURCE_ARN="arn:aws:execute-api:$REGION:$ACCOUNT_ID:$API_ID/*/*"

aws lambda add-permission \
    --function-name "$FUNCTION_NAME" \
    --statement-id "$STATEMENT_ID" \
    --action lambda:InvokeFunction \
    --principal apigateway.amazonaws.com \
    --source-arn "$SOURCE_ARN" || {
    echo "Error: Failed to add Lambda permission" >&2
    exit 1
}

# Deploy API
aws apigateway create-deployment \
    --rest-api-id "$API_ID" \
    --stage-name test \
    --description "Test deployment" || {
    echo "Error: Failed to deploy API" >&2
    exit 1
```

```
}

echo "Testing API..."

# Test the API
INVOKE_URL="https://$API_ID.execute-api.$REGION.amazonaws.com/test/helloworld"

echo "API URL: $INVOKE_URL"

# Test with query parameter (with proper URL encoding)
echo "Testing with query parameter:"
curl -s -X GET "$INVOKE_URL?greeter=John" | jq . 2>/dev/null || curl -s -X GET
 "$INVOKE_URL?greeter=John"
echo ""

# Test with header
echo "Testing with header:"
curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John' | jq . 2>/dev/null || curl -s -X GET "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -H 'greeter: John'
echo ""

# Test with body
echo "Testing with POST body:"
curl -s -X POST "$INVOKE_URL" \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}' | jq . 2>/dev/null || curl -s -X POST "$INVOKE_URL"
 \
    -H 'content-type: application/json' \
    -d '{"greeter": "John"}'
echo ""

echo "Tutorial completed! API is available at: $INVOKE_URL"

# Cleanup
echo "Cleaning up resources..."

# Delete API
aws apigateway delete-rest-api --rest-api-id "$API_ID" || echo "Warning: Failed
 to delete API" >&2

# Delete Lambda function
```

```
aws lambda delete-function --function-name "$FUNCTION_NAME" || echo "Warning:
 Failed to delete Lambda function" >&2

# Detach policy and delete role
aws iam detach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" || echo "Warning: Failed to detach policy" >&2

aws iam delete-role --role-name "$ROLE_NAME" || echo "Warning: Failed to delete
 role" >&2

# Clean up local files securely
rm -f lambda_function.py function.zip trust-policy.json

echo "Cleanup completed!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddPermission
  - AttachRolePolicy
  - CreateDeployment
  - CreateFunction
  - CreateResource
  - CreateRestApi
  - CreateRole
  - DeleteFunction
  - DeleteRestApi
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetResources
  - GetRestApis
  - PutIntegration
  - PutMethod

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Fargate 启动类型的 Amazon ECS Linux 任务

以下代码示例展示了如何：

- 创建集群
- 创建任务定义
- 创建服务
- 清理

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon ECS Fargate Tutorial Script - Version 5
# This script creates an ECS cluster, task definition, and service using Fargate
 launch type
# Fixed version with proper resource dependency handling during cleanup
# Security improvements applied

set -e  # Exit on any error

# Initialize logging
LOG_FILE="ecs-fargate-tutorial-v5.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon ECS Fargate tutorial at $(date)"
echo "Log file: $LOG_FILE"

# Generate random identifier for unique resource names
```

```
RANDOM_ID=$(openssl rand -hex 6)
CLUSTER_NAME="fargate-cluster-${RANDOM_ID}"
SERVICE_NAME="fargate-service-${RANDOM_ID}"
TASK_FAMILY="sample-fargate-${RANDOM_ID}"
SECURITY_GROUP_NAME="ecs-fargate-sg-${RANDOM_ID}"

# Array to track created resources for cleanup
CREATED_RESOURCES=()

# Function to log and execute commands with input validation
execute_command() {
    local cmd="$1"
    local description="$2"

    # Validate that cmd is not empty
    if [[ -z "$cmd" ]]; then
        echo "ERROR: Command is empty"
        return 1
    fi

    echo ""
    echo "==========================================="
    echo "EXECUTING: $description"
    echo "COMMAND: $cmd"
    echo "==========================================="

    local output
    local exit_code
    set +e  # Temporarily disable exit on error
    output=$(eval "$cmd" 2>&1)
    exit_code=$?
    set -e  # Re-enable exit on error

    if [[ $exit_code -eq 0 ]]; then
        echo "SUCCESS: $description"
        echo "OUTPUT: $output"
        return 0
    else
        echo "FAILED: $description"
        echo "EXIT CODE: $exit_code"
        echo "OUTPUT: $output"
        return 1
    fi
}
```

```
# Function to check for actual AWS API errors in command output
check_for_aws_errors() {
    local output="$1"
    local description="$2"

    # Look for specific AWS error patterns, not just the word "error"
    if echo "$output" | grep -qi "An error occurred\|InvalidParameter\|
AccessDenied\|ResourceNotFound\|ValidationException"; then
        echo "AWS API ERROR detected in output for: $description"
        echo "Output: $output"
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
safe_json_extract() {
    local json="$1"
    local key="$2"
    local value

    value=$(echo "$json" | grep -o "\"$key\": \"[^\"]*\"" | cut -d'"' -f4 2>/dev/
null || echo "")
    echo "$value"
}

# Function to wait for network interfaces to be cleaned up
wait_for_network_interfaces_cleanup() {
    local security_group_id="$1"
    local max_attempts=30
    local attempt=1

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    echo "Waiting for network interfaces to be cleaned up..."

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Checking for dependent network
 interfaces..."
```

```
        # Check if there are any network interfaces still using this security
 group
        local eni_count
        eni_count=$(aws ec2 describe-network-interfaces \
            --filters "Name=group-id,Values=$security_group_id" \
            --query "length(NetworkInterfaces)" \
            --output text 2>/dev/null || echo "0")

        if [[ "$eni_count" == "0" ]]; then
            echo "No network interfaces found using security group
 $security_group_id"
            return 0
        else
            echo "Found $eni_count network interface(s) still using security
 group $security_group_id"
            echo "Waiting 10 seconds before next check..."
            sleep 10
            ((attempt++))
        fi
    done

    echo "WARNING: Network interfaces may still be attached after $max_attempts
 attempts"
    echo "This is normal and the security group deletion will be retried"
    return 1
}

# Function to retry security group deletion with exponential backoff
retry_security_group_deletion() {
    local security_group_id="$1"
    local max_attempts=10
    local attempt=1
    local wait_time=5

    # Validate security group ID format
    if [[ ! "$security_group_id" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
        echo "ERROR: Invalid security group ID format: $security_group_id"
        return 1
    fi

    while [[ $attempt -le $max_attempts ]]; do
        echo "Attempt $attempt/$max_attempts: Trying to delete security group
 $security_group_id"
```

```
        if execute_command "aws ec2 delete-security-group --group-id
 '$security_group_id'" "Delete security group (attempt $attempt)"; then
            echo "Successfully deleted security group $security_group_id"
            return 0
        else
            if [[ $attempt -eq $max_attempts ]]; then
                echo "FAILED: Could not delete security group $security_group_id
 after $max_attempts attempts"
                echo "This may be due to network interfaces that are still being
 cleaned up by AWS"
                echo "You can manually delete it later using: aws ec2 delete-
security-group --group-id $security_group_id"
                return 1
            else
                echo "Waiting $wait_time seconds before retry..."
                sleep $wait_time
                wait_time=$((wait_time * 2))  # Exponential backoff
                ((attempt++))
            fi
        fi
    done
}

# Function to cleanup resources with proper dependency handling
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "CLEANUP PROCESS"
    echo "==========================================="
    echo "The following resources were created:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "Auto-confirming cleanup of all created resources..."

    CLEANUP_CHOICE="y"

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup process..."

        # Step 1: Scale service to 0 tasks first, then delete service
```

```
       if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Service: $SERVICE_NAME " ]];
then
           echo ""
           echo "Step 1: Scaling service to 0 tasks..."
           if execute_command "aws ecs update-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --desired-count 0" "Scale service to 0 tasks"; then
               echo "Waiting for service to stabilize after scaling to 0..."
               execute_command "aws ecs wait services-stable --cluster
'$CLUSTER_NAME' --services '$SERVICE_NAME'" "Wait for service to stabilize"

               echo "Deleting service..."
               execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME'" "Delete ECS service"
           else
               echo "WARNING: Failed to scale service. Attempting to delete
anyway..."
               execute_command "aws ecs delete-service --cluster '$CLUSTER_NAME'
--service '$SERVICE_NAME' --force" "Force delete ECS service"
           fi
       fi

       # Step 2: Wait a bit for tasks to fully terminate
       echo ""
       echo "Step 2: Waiting for tasks to fully terminate..."
       sleep 15

       # Step 3: Delete cluster
       if [[ " ${CREATED_RESOURCES[*]} " =~ " ECS Cluster: $CLUSTER_NAME " ]];
then
           echo ""
           echo "Step 3: Deleting cluster..."
           execute_command "aws ecs delete-cluster --cluster '$CLUSTER_NAME'"
"Delete ECS cluster"
       fi

       # Step 4: Wait for network interfaces to be cleaned up, then delete
security group
       if [[ -n "$SECURITY_GROUP_ID" && "$SECURITY_GROUP_ID" != "None" ]]; then
           echo ""
           echo "Step 4: Cleaning up security group..."

           # First, wait for network interfaces to be cleaned up
           wait_for_network_interfaces_cleanup "$SECURITY_GROUP_ID"
```

```
            # Then retry security group deletion with backoff
            retry_security_group_deletion "$SECURITY_GROUP_ID"
        fi

        # Step 5: Clean up task definition (deregister all revisions)
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo ""
            echo "Step 5: Deregistering task definition revisions..."

            # Get all revisions of the task definition
            local revisions
            revisions=$(aws ecs list-task-definitions --family-prefix
 "$TASK_FAMILY" --query "taskDefinitionArns" --output text 2>/dev/null || echo
 "")

            if [[ -n "$revisions" && "$revisions" != "None" ]]; then
                for revision_arn in $revisions; do
                    echo "Deregistering task definition: $revision_arn"
                    execute_command "aws ecs deregister-task-definition --task-
definition '$revision_arn'" "Deregister task definition $revision_arn" || true
                done
            else
                echo "No task definition revisions found to deregister"
            fi
        fi

        echo ""
        echo "============================================"
        echo "CLEANUP COMPLETED"
        echo "============================================"
        echo "All resources have been cleaned up successfully!"

    else
        echo "Cleanup skipped. Resources remain active."
        echo ""
        echo "To clean up manually later, use the following commands in order:"
        echo "1. Scale service to 0: aws ecs update-service --cluster
 $CLUSTER_NAME --service $SERVICE_NAME --desired-count 0"
        echo "2. Wait for stability: aws ecs wait services-stable --cluster
 $CLUSTER_NAME --services $SERVICE_NAME"
        echo "3. Delete service: aws ecs delete-service --cluster $CLUSTER_NAME
 --service $SERVICE_NAME"
        echo "4. Delete cluster: aws ecs delete-cluster --cluster $CLUSTER_NAME"
```

```
        echo "5. Wait 2-3 minutes, then delete security group: aws ec2 delete-
security-group --group-id $SECURITY_GROUP_ID"
        if [[ " ${CREATED_RESOURCES[*]} " =~ " Task Definition: $TASK_FAMILY
 " ]]; then
            echo "6. Deregister task definitions: aws ecs list-task-definitions
 --family-prefix $TASK_FAMILY"
            echo "   Then for each ARN: aws ecs deregister-task-definition --
task-definition <ARN>"
        fi
    fi
}

# Trap to handle script interruption
trap cleanup_resources EXIT

echo "Using random identifier: $RANDOM_ID"
echo "Cluster name: $CLUSTER_NAME"
echo "Service name: $SERVICE_NAME"
echo "Task family: $TASK_FAMILY"

# Step 1: Ensure ECS task execution role exists
echo ""
echo "==========================================="
echo "STEP 1: VERIFY ECS TASK EXECUTION ROLE"
echo "==========================================="

ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ ! "$ACCOUNT_ID" =~ ^[0-9]{12}$ ]]; then
    echo "ERROR: Invalid AWS Account ID retrieved: $ACCOUNT_ID"
    exit 1
fi

EXECUTION_ROLE_ARN="arn:aws:iam::${ACCOUNT_ID}:role/ecsTaskExecutionRole"

# Check if role exists
if aws iam get-role --role-name ecsTaskExecutionRole >/dev/null 2>&1; then
    echo "ECS task execution role already exists"
else
    echo "Creating ECS task execution role..."

    # Create trust policy with strict validation
    cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
```

```
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ecs-tasks.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF

    # Validate JSON before using
    if ! jq empty trust-policy.json 2>/dev/null; then
        echo "ERROR: Invalid JSON in trust policy"
        rm -f trust-policy.json
        exit 1
    fi

    execute_command "aws iam create-role --role-name ecsTaskExecutionRole --
assume-role-policy-document file://trust-policy.json" "Create ECS task execution
 role"

    aws iam tag-role --role-name ecsTaskExecutionRole --tags
 Key=project,Value=doc-smith Key=tutorial,Value=amazon-ecs-fargate-linux

    execute_command "aws iam attach-role-policy --role-name
 ecsTaskExecutionRole --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonECSTaskExecutionRolePolicy" "Attach ECS task execution policy"

    # Clean up temporary file securely
    shred -vfz -n 3 trust-policy.json 2>/dev/null || rm -f trust-policy.json

    CREATED_RESOURCES+=("IAM Role: ecsTaskExecutionRole")
fi

# Step 2: Create ECS cluster
echo ""
echo "============================================="
echo "STEP 2: CREATE ECS CLUSTER"
echo "============================================="
```

```
CLUSTER_OUTPUT=$(execute_command "aws ecs create-cluster --cluster-name
 '$CLUSTER_NAME' --tags key=project,value=doc-smith key=tutorial,value=amazon-
ecs-fargate-linux" "Create ECS cluster")
check_for_aws_errors "$CLUSTER_OUTPUT" "Create ECS cluster"

CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")

# Step 3: Create task definition
echo ""
echo "============================================="
echo "STEP 3: CREATE TASK DEFINITION"
echo "============================================="

# Create task definition JSON with validated inputs
cat > task-definition.json << EOF
{
    "family": "$TASK_FAMILY",
    "networkMode": "awsvpc",
    "requiresCompatibilities": ["FARGATE"],
    "cpu": "256",
    "memory": "512",
    "executionRoleArn": "$EXECUTION_ROLE_ARN",
    "containerDefinitions": [
        {
            "name": "fargate-app",
            "image": "public.ecr.aws/docker/library/httpd:2.4-alpine",
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "essential": true,
            "entryPoint": ["sh", "-c"],
            "command": [
                "/bin/sh -c \"echo '<html> <head> <title>Amazon ECS Sample
 App</title> <style>body {margin-top: 40px; background-color: #333;} </style> </
head><body> <div style=color:white;text-align:center> <h1>Amazon ECS Sample App</
h1> <h2>Congratulations!</h2> <p>Your application is now running on a container
 in Amazon ECS.</p> </div></body></html>' >  /usr/local/apache2/htdocs/index.html
 && httpd-foreground\""
            ],
            "logConfiguration": {
```

```
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/fargate-sample",
                    "awslogs-region": "us-east-1",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ]
}
EOF

# Validate JSON before using
if ! jq empty task-definition.json 2>/dev/null; then
    echo "ERROR: Invalid JSON in task definition"
    rm -f task-definition.json
    exit 1
fi

TASK_DEF_OUTPUT=$(execute_command "aws ecs register-task-definition --cli-input-
json file://task-definition.json" "Register task definition")
check_for_aws_errors "$TASK_DEF_OUTPUT" "Register task definition"

# Clean up temporary file securely
shred -vfz -n 3 task-definition.json 2>/dev/null || rm -f task-definition.json

CREATED_RESOURCES+=("Task Definition: $TASK_FAMILY")

# Step 4: Set up networking
echo ""
echo "============================================="
echo "STEP 4: SET UP NETWORKING"
echo "============================================="

# Get default VPC ID
VPC_ID=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true" --query
 "Vpcs[0].VpcId" --output text)
if [[ "$VPC_ID" == "None" || -z "$VPC_ID" ]]; then
    echo "ERROR: No default VPC found. Please create a default VPC or specify a
 custom VPC."
    exit 1
fi

# Validate VPC ID format
```

```
if [[ ! "$VPC_ID" =~ ^vpc-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid VPC ID format: $VPC_ID"
    exit 1
fi

echo "Using default VPC: $VPC_ID"

# Create security group with restricted access
# Note: This allows HTTP access from anywhere for demo purposes
# In production, restrict source to specific IP ranges or security groups
SECURITY_GROUP_OUTPUT=$(execute_command "aws ec2 create-security-group --
group-name '$SECURITY_GROUP_NAME' --description 'Security group for ECS
 Fargate tutorial - HTTP access' --vpc-id '$VPC_ID' --tag-specifications
 'ResourceType=security-group,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-ecs-fargate-linux}]'" "Create security group")
check_for_aws_errors "$SECURITY_GROUP_OUTPUT" "Create security group"

SECURITY_GROUP_ID=$(echo "$SECURITY_GROUP_OUTPUT" | grep -o '"GroupId": "[^"]*"'
 | head -1 | cut -d'"' -f4)
if [[ -z "$SECURITY_GROUP_ID" || "$SECURITY_GROUP_ID" == "None" ]]; then
    SECURITY_GROUP_ID=$(aws ec2 describe-security-groups --group-names
 "$SECURITY_GROUP_NAME" --filters "Name=vpc-id,Values=$VPC_ID" --query
 "SecurityGroups[0].GroupId" --output text)
fi

# Validate security group ID format
if [[ ! "$SECURITY_GROUP_ID" =~ ^sg-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid security group ID format: $SECURITY_GROUP_ID"
    exit 1
fi

echo "Created security group: $SECURITY_GROUP_ID"
CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")

# Add HTTP inbound rule
# WARNING: This allows HTTP access from anywhere (0.0.0.0/0)
# In production environments, restrict this to specific IP ranges
execute_command "aws ec2 authorize-security-group-ingress --group-id
 '$SECURITY_GROUP_ID' --protocol tcp --port 80 --cidr 0.0.0.0/0" "Add HTTP
 inbound rule to security group"

# Get subnet IDs from default VPC
echo "Getting subnet IDs from default VPC..."
```

```
SUBNET_IDS_RAW=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=$VPC_ID"
 --query "Subnets[*].SubnetId" --output text)
if [[ -z "$SUBNET_IDS_RAW" ]]; then
    echo "ERROR: No subnets found in default VPC"
    exit 1
fi

# Convert to proper comma-separated format, handling both spaces and tabs
SUBNET_IDS_COMMA=$(echo "$SUBNET_IDS_RAW" | tr -s '[:space:]' ',' | sed 's/,$//')
echo "Raw subnet IDs: $SUBNET_IDS_RAW"
echo "Formatted subnet IDs: $SUBNET_IDS_COMMA"

# Validate subnet IDs format
if [[ ! "$SUBNET_IDS_COMMA" =~ ^subnet-[a-z0-9]+(,subnet-[a-z0-9]+)*$ ]]; then
    echo "ERROR: Invalid subnet ID format: $SUBNET_IDS_COMMA"
    exit 1
fi

# Step 5: Create ECS service
echo ""
echo "=========================================="
echo "STEP 5: CREATE ECS SERVICE"
echo "=========================================="

# Create the service with proper JSON formatting for network configuration
SERVICE_CMD="aws ecs create-service --cluster '$CLUSTER_NAME' --service-name
 '$SERVICE_NAME' --task-definition '$TASK_FAMILY' --desired-count 1 --launch-
type FARGATE --network-configuration '{\"awsvpcConfiguration\":{\"subnets
\":[\"$(echo "$SUBNET_IDS_COMMA" | sed 's/,/","/g')\"],\"securityGroups
\":[\"$SECURITY_GROUP_ID\"],\"assignPublicIp\":\"ENABLED\"}}' --tags
 key=project,value=doc-smith key=tutorial,value=amazon-ecs-fargate-linux"

echo "Service creation command: $SERVICE_CMD"

SERVICE_OUTPUT=$(execute_command "$SERVICE_CMD" "Create ECS service")
check_for_aws_errors "$SERVICE_OUTPUT" "Create ECS service"

CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

# Step 6: Wait for service to stabilize and get public IP
echo ""
echo "=========================================="
echo "STEP 6: WAIT FOR SERVICE AND GET PUBLIC IP"
echo "=========================================="
```

```
echo "Waiting for service to stabilize (this may take a few minutes)..."
execute_command "aws ecs wait services-stable --cluster '$CLUSTER_NAME' --
services '$SERVICE_NAME'" "Wait for service to stabilize"

# Get task ARN
TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query "taskArns[0]" --output text)
if [[ "$TASK_ARN" == "None" || -z "$TASK_ARN" ]]; then
    echo "ERROR: No running tasks found for service"
    exit 1
fi

echo "Task ARN: $TASK_ARN"

# Get network interface ID
ENI_ID=$(aws ecs describe-tasks --cluster "$CLUSTER_NAME" --tasks "$TASK_ARN"
 --query "tasks[0].attachments[0].details[?name=='networkInterfaceId'].value" --
output text)
if [[ "$ENI_ID" == "None" || -z "$ENI_ID" ]]; then
    echo "ERROR: Could not retrieve network interface ID"
    exit 1
fi

# Validate ENI ID format
if [[ ! "$ENI_ID" =~ ^eni-[a-z0-9]{8,17}$ ]]; then
    echo "ERROR: Invalid network interface ID format: $ENI_ID"
    exit 1
fi

echo "Network Interface ID: $ENI_ID"

# Get public IP
PUBLIC_IP=$(aws ec2 describe-network-interfaces --network-interface-ids "$ENI_ID"
 --query "NetworkInterfaces[0].Association.PublicIp" --output text)
if [[ "$PUBLIC_IP" == "None" || -z "$PUBLIC_IP" ]]; then
    echo "WARNING: No public IP assigned to the task"
    echo "The task may be in a private subnet or public IP assignment failed"
else
    # Validate IP format
    if [[ ! "$PUBLIC_IP" =~ ^[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}$ ]];
 then
        echo "ERROR: Invalid IP address format: $PUBLIC_IP"
        PUBLIC_IP=""
```

```
    else
        echo ""
        echo "==========================================="
        echo "SUCCESS! APPLICATION IS RUNNING"
        echo "==========================================="
        echo "Your application is available at: http://$PUBLIC_IP"
        echo "You can test it by opening this URL in your browser"
        echo ""
    fi
fi

# Display service information
echo ""
echo "==========================================="
echo "SERVICE INFORMATION"
echo "==========================================="
execute_command "aws ecs describe-services --cluster '$CLUSTER_NAME' --services
 '$SERVICE_NAME'" "Get service details"

echo ""
echo "==========================================="
echo "TUTORIAL COMPLETED SUCCESSFULLY"
echo "==========================================="
echo "Resources created:"
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "  - $resource"
done

if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
    echo ""
    echo "Application URL: http://$PUBLIC_IP"
fi

echo ""
echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateRole

- CreateSecurityGroup
- CreateService
- DeleteCluster
- DeleteSecurityGroup
- DeleteService
- DeregisterTaskDefinition
- DescribeNetworkInterfaces
- DescribeSecurityGroups
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetRole
- ListTaskDefinitions
- ListTasks
- RegisterTaskDefinition
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将函数名称作为变量创建 CloudWatch 控制面板

以下代码示例展示了如何：

- 创建 CloudWatch 控制面板
- 添加带有函数名称变量的 Lambda 指标小组件
- 验证控制面板
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Script to create a CloudWatch dashboard with Lambda function name as a variable
# This script creates a CloudWatch dashboard that allows you to switch between
 different Lambda functions

# Set up logging with secure permissions
LOG_FILE="${HOME}/.cloudwatch-dashboard-script.log"
touch "$LOG_FILE" && chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "$(date): Starting CloudWatch dashboard creation script"

# Security: Set strict error handling
set -uo pipefail
trap 'handle_error "Script failed at line $LINENO"' ERR

# Function to handle errors
handle_error() {
    local error_msg="${1:-Unknown error}"
    echo "ERROR: $error_msg" >&2
    echo "Resources created:"
    echo "- CloudWatch Dashboard: LambdaMetricsDashboard"
    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "An error occurred. Proceeding with automatic cleanup..."

    echo "Cleaning up resources..."
    aws cloudwatch delete-dashboards --dashboard-names LambdaMetricsDashboard 2>/
dev/null || true
```

```
    # Clean up temporary files
    if [ -n "${TEMP_DIR:-}" ] && [ -d "$TEMP_DIR" ]; then
        rm -rf "$TEMP_DIR"
    fi
    rm -f dashboard-body.json

    echo "Cleanup complete."
    exit 1
}

# Security: Validate AWS CLI is installed
if ! command -v aws &> /dev/null; then
    handle_error "AWS CLI is not installed. Please install it and try again."
fi

# Check if AWS CLI is installed and configured
echo "Checking AWS CLI configuration..."
if ! aws sts get-caller-identity > /dev/null 2>&1; then
    handle_error "AWS CLI is not properly configured. Please configure it with
 'aws configure' and try again."
fi

# Get the current region securely
REGION=$(aws configure get region 2>/dev/null || echo "")
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $REGION"
fi
echo "Using region: $REGION"

# Validate region format
if ! [[ "$REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]{1}$ ]]; then
    handle_error "Invalid AWS region format: $REGION"
fi

# Check if there are any Lambda functions in the account
echo "Checking for Lambda functions..."
LAMBDA_FUNCTIONS=$(aws lambda list-functions --region "$REGION" --query
 "Functions[*].FunctionName" --output text 2>/dev/null || echo "")

if [ -z "$LAMBDA_FUNCTIONS" ]; then
    echo "No Lambda functions found in your account. Creating a simple test
 function..."
```

```
    # Create a temporary directory for Lambda function code with secure
 permissions
    TEMP_DIR=$(mktemp -d)
    chmod 700 "$TEMP_DIR"
    trap 'rm -rf "$TEMP_DIR"' EXIT

    # Create a simple Lambda function
    cat > "$TEMP_DIR/index.js" << 'EOF'
exports.handler = async (event) => {
    console.log('Event:', JSON.stringify(event, null, 2));
    return {
        statusCode: 200,
        body: JSON.stringify('Hello from Lambda!'),
    };
};
EOF

    # Zip the function code
    if ! cd "$TEMP_DIR"; then
        handle_error "Failed to change to temporary directory"
    fi

    if ! zip -q function.zip index.js; then
        handle_error "Failed to create zip file"
    fi

    # Create a role for the Lambda function with restricted trust policy
    ROLE_NAME="LambdaDashboardTestRole-$(date +%s)"
    TRUST_POLICY='{"Version":"2012-10-17",       "Statement":
[{"Effect":"Allow","Principal":
{"Service":"lambda.amazonaws.com"},"Action":"sts:AssumeRole"}]}'

    if ! ROLE_ARN=$(aws iam create-role \
        --role-name "$ROLE_NAME" \
        --assume-role-policy-document "$TRUST_POLICY" \
        --tags Key=project,Value=doc-smith Key=tutorial,Value=cloudwatch-
dynamicdash \
        --query "Role.Arn" \
        --output text 2>/dev/null); then
        handle_error "Failed to create IAM role for Lambda function"
    fi

    echo "Waiting for role to be available..."
```

```
    sleep 10

    # Attach basic Lambda execution policy
    if ! aws iam attach-role-policy \
        --role-name "$ROLE_NAME" \
        --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole"; then
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to attach policy to IAM role"
    fi

    # Create the Lambda function
    FUNCTION_NAME="DashboardTestFunction-$(date +%s)"
    if ! aws lambda create-function \
        --function-name "$FUNCTION_NAME" \
        --runtime nodejs18.x \
        --role "$ROLE_ARN" \
        --handler index.handler \
        --zip-file fileb://function.zip \
        --tags project=doc-smith,tutorial=cloudwatch-dynamicdash \
        --region "$REGION" > /dev/null 2>&1; then
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
        handle_error "Failed to create Lambda function"
    fi

    # Invoke the function to generate some metrics
    echo "Invoking Lambda function to generate metrics..."
    for i in {1..5}; do
        aws lambda invoke --function-name "$FUNCTION_NAME" --payload '{}' /dev/
null --region "$REGION" > /dev/null 2>&1 || true
        sleep 1
    done

    # Go back to original directory
    cd - > /dev/null

    # Set the function name for the dashboard
    DEFAULT_FUNCTION="$FUNCTION_NAME"
else
    # Use the first Lambda function as default
```

```
    DEFAULT_FUNCTION=$(echo "$LAMBDA_FUNCTIONS" | awk '{print $1}')
    echo "Found Lambda functions. Using $DEFAULT_FUNCTION as default."
    FUNCTION_NAME=""
    ROLE_NAME=""
fi

# Create a dashboard with Lambda metrics and a function name variable
echo "Creating CloudWatch dashboard with Lambda function name variable..."

# Create a JSON file for the dashboard body with secure permissions
DASHBOARD_JSON="dashboard-body-$$.json"
touch "$DASHBOARD_JSON" && chmod 600 "$DASHBOARD_JSON"

# Escape special characters in region and function name for JSON
REGION_ESCAPED=$(printf '%s\n' "$REGION" | sed 's:[\/&]:\\&:g')
FUNCTION_ESCAPED=$(printf '%s\n' "$DEFAULT_FUNCTION" | sed 's:[\/&]:\\&:g')

cat > "$DASHBOARD_JSON" << EOF
{
  "widgets": [
    {
      "type": "metric",
      "x": 0,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Invocations", "FunctionName", "\${FunctionName}" ],
          [ ".", "Errors", ".", "." ],
          [ ".", "Throttles", ".", "." ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Lambda Function Metrics for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 0,
      "y": 6,
      "width": 12,
```

```
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "Duration", "FunctionName", "\${FunctionName}",
 { "stat": "Average" } ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Duration for \${FunctionName}",
        "period": 300
      }
    },
    {
      "type": "metric",
      "x": 12,
      "y": 0,
      "width": 12,
      "height": 6,
      "properties": {
        "metrics": [
          [ "AWS/Lambda", "ConcurrentExecutions", "FunctionName", "\
${FunctionName}" ]
        ],
        "view": "timeSeries",
        "stacked": false,
        "region": "$REGION_ESCAPED",
        "title": "Concurrent Executions for \${FunctionName}",
        "period": 300
      }
    }
  ],
  "periodOverride": "auto",
  "variables": [
    {
      "type": "property",
      "id": "FunctionName",
      "property": "FunctionName",
      "label": "Lambda Function",
      "inputType": "select",
      "values": [
        {
          "value": "$FUNCTION_ESCAPED",
          "label": "$FUNCTION_ESCAPED"
```

```
        }
      ]
    }
  ]
}
EOF

# Validate JSON before sending
if ! jq empty "$DASHBOARD_JSON" 2>/dev/null; then
    handle_error "Invalid JSON generated for dashboard"
fi

# Create the dashboard using the JSON file
DASHBOARD_NAME="LambdaMetricsDashboard-$(date +%s)"
if ! DASHBOARD_RESULT=$(aws cloudwatch put-dashboard \
    --dashboard-name "$DASHBOARD_NAME" \
    --dashboard-body file://"$DASHBOARD_JSON" \
    --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to create CloudWatch dashboard."
fi

# Display any validation messages but continue
if echo "$DASHBOARD_RESULT" | grep -q "DashboardValidationMessages"; then
    echo "Dashboard created with validation messages:"
    echo "$DASHBOARD_RESULT"
    echo "These validation messages are warnings and the dashboard should still
 function."
else
    echo "Dashboard created successfully!"
fi

# Verify the dashboard was created
echo "Verifying dashboard creation..."
```

```
if ! DASHBOARD_INFO=$(aws cloudwatch get-dashboard --dashboard-name
 "$DASHBOARD_NAME" --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to verify dashboard creation."
fi

echo "Dashboard verification successful!"
echo "Dashboard details:"
echo "$DASHBOARD_INFO"

# List all dashboards to confirm
echo "Listing all dashboards:"
if ! DASHBOARDS=$(aws cloudwatch list-dashboards --region "$REGION" 2>&1); then
    # If we created resources, clean them up
    if [ -n "${FUNCTION_NAME:-}" ]; then
        aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null || true
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null || true
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
    handle_error "Failed to list dashboards."
fi
echo "$DASHBOARDS"

# Show instructions for accessing the dashboard
echo ""
echo "Dashboard created successfully! To access it:"
echo "1. Open the CloudWatch console at https://console.aws.amazon.com/
cloudwatch/"
echo "2. In the navigation pane, choose Dashboards"
echo "3. Select $DASHBOARD_NAME"
```

```
echo "4. You should see a dropdown menu labeled 'Lambda Function' at the top of
 the dashboard"
echo "5. Use this dropdown to select different Lambda functions and see their
 metrics"
echo ""

# Create a list of resources for cleanup
RESOURCES=("- CloudWatch Dashboard: $DASHBOARD_NAME")
if [ -n "${FUNCTION_NAME:-}" ]; then
    RESOURCES+=("- Lambda Function: $FUNCTION_NAME")
    RESOURCES+=("- IAM Role: $ROLE_NAME")
fi

# Prompt for cleanup with automatic yes
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
for resource in "${RESOURCES[@]}"; do
    echo "$resource"
done
echo ""
echo "Proceeding with automatic cleanup..."

CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."

    # Delete the dashboard
    if aws cloudwatch delete-dashboards --dashboard-names "$DASHBOARD_NAME" --
region "$REGION" 2>/dev/null; then
        echo "Dashboard deleted successfully."
    else
        echo "WARNING: Failed to delete dashboard. You may need to delete it
 manually."
    fi

    # If we created a Lambda function, delete it and its role
    if [ -n "${FUNCTION_NAME:-}" ]; then
        echo "Deleting Lambda function..."
        if aws lambda delete-function --function-name "$FUNCTION_NAME" --region
 "$REGION" 2>/dev/null; then
            echo "Lambda function deleted successfully."
```

```
        else
            echo "WARNING: Failed to delete Lambda function. You may need to
 delete it manually."
        fi

        echo "Detaching role policy..."
        if aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "arn:aws:iam::aws:policy/service-role/
AWSLambdaBasicExecutionRole" 2>/dev/null; then
            echo "Role policy detached successfully."
        else
            echo "WARNING: Failed to detach role policy. You may need to detach
 it manually."
        fi

        echo "Deleting IAM role..."
        if aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null; then
            echo "IAM role deleted successfully."
        else
            echo "WARNING: Failed to delete IAM role. You may need to delete it
 manually."
        fi
    fi

    # Clean up the JSON file
    rm -f "$DASHBOARD_JSON"

    echo "Cleanup complete."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateFunction
  - CreateRole
  - DeleteDashboards
  - DeleteFunction
  - DeleteRole

- DetachRolePolicy
- GetCallerIdentity
- GetDashboard
- Invoke
- ListDashboards
- ListFunctions
- PutDashboard

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 EC2 启动类型的 Amazon ECS 服务

以下代码示例展示了如何：

- 创建 ECS 集群
- 创建和监控服务
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# ECS EC2 Launch Type Tutorial Script - UPDATED VERSION
# This script demonstrates creating an ECS cluster, launching a container
 instance,
# registering a task definition, and creating a service using the EC2 launch
 type.
# Updated to match the tutorial draft with nginx web server and service creation.
```

```
#
# - UPDATED: Changed from sleep task to nginx web server with service

set -e  # Exit on any error

# Configuration
SCRIPT_NAME="ecs-ec2-tutorial"
LOG_FILE="${SCRIPT_NAME}-$(date +%Y%m%d-%H%M%S).log"
CLUSTER_NAME="tutorial-cluster-$(openssl rand -hex 4)"
TASK_FAMILY="nginx-task-$(openssl rand -hex 4)"
SERVICE_NAME="nginx-service-$(openssl rand -hex 4)"
KEY_PAIR_NAME="ecs-tutorial-key-$(openssl rand -hex 4)"
SECURITY_GROUP_NAME="ecs-tutorial-sg-$(openssl rand -hex 4)"

# Tags
PROJECT_TAG="doc-smith"
TUTORIAL_TAG="ecs-ec2"

# Get current AWS region dynamically
AWS_REGION=$(aws configure get region || echo "us-east-1")

# Resource tracking arrays
CREATED_RESOURCES=()
CLEANUP_ORDER=()

# Logging function
log() {
    echo "[$(date '+%Y-%m-%d %H:%M:%S')] $1" | tee -a "$LOG_FILE"
}

# Error handling function
handle_error() {
    local exit_code=$?
    log "ERROR: Script failed with exit code $exit_code"
    log "ERROR: Last command: $BASH_COMMAND"

    echo ""
    echo "========================================="
    echo "ERROR OCCURRED - ATTEMPTING CLEANUP"
    echo "========================================="
    echo "Resources created before error:"
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
```

```
    cleanup_resources
    exit $exit_code
}

# Set error trap
trap handle_error ERR

# FIXED: Enhanced cleanup function with proper error handling and logging
cleanup_resources() {
    log "Starting cleanup process..."
    local cleanup_errors=0

    # Delete service first (this will stop tasks automatically)
    if [[ -n "${SERVICE_ARN:-}" ]]; then
        log "Updating service to desired count 0: $SERVICE_NAME"
        if ! aws ecs update-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" --desired-count 0 2>>"$LOG_FILE"; then
            log "WARNING: Failed to update service desired count to 0"
            ((cleanup_errors++))
        else
            log "Waiting for service tasks to stop..."
            sleep 30  # Give time for tasks to stop
        fi

        log "Deleting service: $SERVICE_NAME"
        if ! aws ecs delete-service --cluster "$CLUSTER_NAME" --service
 "$SERVICE_NAME" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to delete service $SERVICE_NAME"
            ((cleanup_errors++))
        fi
    fi

    # Stop and delete any remaining tasks
    if [[ -n "${TASK_ARN:-}" ]]; then
        log "Stopping task: $TASK_ARN"
        if ! aws ecs stop-task --cluster "$CLUSTER_NAME" --task "$TASK_ARN" --
reason "Tutorial cleanup" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to stop task $TASK_ARN"
            ((cleanup_errors++))
        else
            log "Waiting for task to stop..."
            if ! aws ecs wait tasks-stopped --cluster "$CLUSTER_NAME" --tasks
 "$TASK_ARN" 2>>"$LOG_FILE"; then
```

```
                log "WARNING: Task stop wait failed for $TASK_ARN"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Deregister task definition
    if [[ -n "${TASK_DEFINITION_ARN:-}" ]]; then
        log "Deregistering task definition: $TASK_DEFINITION_ARN"
        if ! aws ecs deregister-task-definition --task-definition
"$TASK_DEFINITION_ARN" 2>>"$LOG_FILE"; then
            log "WARNING: Failed to deregister task definition
$TASK_DEFINITION_ARN"
            ((cleanup_errors++))
        fi
    fi

    # Terminate EC2 instance
    if [[ -n "${INSTANCE_ID:-}" ]]; then
        log "Terminating EC2 instance: $INSTANCE_ID"
        if ! aws ec2 terminate-instances --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
            log "WARNING: Failed to terminate instance $INSTANCE_ID"
            ((cleanup_errors++))
        else
            log "Waiting for instance to terminate..."
            if ! aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"
2>>"$LOG_FILE"; then
                log "WARNING: Instance termination wait failed for $INSTANCE_ID"
                ((cleanup_errors++))
            fi
        fi
    fi

    # Delete security group with retry logic
    if [[ -n "${SECURITY_GROUP_ID:-}" ]]; then
        log "Deleting security group: $SECURITY_GROUP_ID"
        local retry_count=0
        local max_retries=3

        while [[ $retry_count -lt $max_retries ]]; do
            if aws ec2 delete-security-group --group-id "$SECURITY_GROUP_ID"
2>>"$LOG_FILE"; then
                log "Successfully deleted security group"
```

```
                break
            else
                ((retry_count++))
                if [[ $retry_count -lt $max_retries ]]; then
                    log "Retry $retry_count/$max_retries: Waiting 10 seconds
before retrying security group deletion..."
                    sleep 10
                else
                    log "ERROR: Failed to delete security group after
$max_retries attempts"
                    ((cleanup_errors++))
                fi
            fi
        done
    fi

    # Delete key pair
    if [[ -n "${KEY_PAIR_NAME:-}" ]]; then
        log "Deleting key pair: $KEY_PAIR_NAME"
        if ! aws ec2 delete-key-pair --key-name "$KEY_PAIR_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete key pair $KEY_PAIR_NAME"
            ((cleanup_errors++))
        fi
        rm -f "${KEY_PAIR_NAME}.pem" 2>>"$LOG_FILE" || log "WARNING: Failed to
remove local key file"
    fi

    # Delete ECS cluster
    if [[ -n "${CLUSTER_NAME:-}" ]]; then
        log "Deleting ECS cluster: $CLUSTER_NAME"
        if ! aws ecs delete-cluster --cluster "$CLUSTER_NAME" 2>>"$LOG_FILE";
then
            log "WARNING: Failed to delete cluster $CLUSTER_NAME"
            ((cleanup_errors++))
        fi
    fi

    if [[ $cleanup_errors -eq 0 ]]; then
        log "Cleanup completed successfully"
    else
        log "Cleanup completed with $cleanup_errors warnings/errors. Check log
file for details."
    fi
```

```
}

# Function to check prerequisites
check_prerequisites() {
    log "Checking prerequisites..."

    # Check AWS CLI
    if ! command -v aws &> /dev/null; then
        log "ERROR: AWS CLI is not installed"
        exit 1
    fi

    # Check AWS credentials
    if ! aws sts get-caller-identity &> /dev/null; then
        log "ERROR: AWS credentials not configured"
        exit 1
    fi

    # Get caller identity
    CALLER_IDENTITY=$(aws sts get-caller-identity --output text --query
 'Account')
    log "AWS Account: $CALLER_IDENTITY"
    log "AWS Region: $AWS_REGION"

    # Check for default VPC
    DEFAULT_VPC=$(aws ec2 describe-vpcs --filters "Name=is-default,Values=true"
 --query 'Vpcs[0].VpcId' --output text)
    if [[ "$DEFAULT_VPC" == "None" ]]; then
        log "ERROR: No default VPC found. Please create a VPC first."
        exit 1
    fi
    log "Using default VPC: $DEFAULT_VPC"

    # Get default subnet
    DEFAULT_SUBNET=$(aws ec2 describe-subnets --filters "Name=vpc-id,Values=
$DEFAULT_VPC" "Name=default-for-az,Values=true" --query 'Subnets[0].SubnetId' --
output text)
    if [[ "$DEFAULT_SUBNET" == "None" ]]; then
        log "ERROR: No default subnet found"
        exit 1
    fi
    log "Using default subnet: $DEFAULT_SUBNET"

    log "Prerequisites check completed successfully"
```

```
}

# Function to create ECS cluster
create_cluster() {
    log "Creating ECS cluster: $CLUSTER_NAME"

    CLUSTER_ARN=$(aws ecs create-cluster --cluster-name "$CLUSTER_NAME" --
tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG --query
 'cluster.clusterArn' --output text)

    if [[ -z "$CLUSTER_ARN" ]]; then
        log "ERROR: Failed to create cluster"
        exit 1
    fi

    log "Created cluster: $CLUSTER_ARN"
    CREATED_RESOURCES+=("ECS Cluster: $CLUSTER_NAME")
}

# Function to create key pair
create_key_pair() {
    log "Creating EC2 key pair: $KEY_PAIR_NAME"

    # FIXED: Set secure umask before key creation
    umask 077
    aws ec2 create-key-pair --key-name "$KEY_PAIR_NAME" --query 'KeyMaterial' --
output text > "${KEY_PAIR_NAME}.pem"
    chmod 400 "${KEY_PAIR_NAME}.pem"
    umask 022  # Reset umask

    log "Created key pair: $KEY_PAIR_NAME"
    CREATED_RESOURCES+=("EC2 Key Pair: $KEY_PAIR_NAME")

    aws ec2 create-tags --resources "$KEY_PAIR_NAME" --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG 2>>"$LOG_FILE" || log "WARNING:
 Failed to tag key pair"
}

# Function to create security group
create_security_group() {
    log "Creating security group: $SECURITY_GROUP_NAME"

    SECURITY_GROUP_ID=$(aws ec2 create-security-group \
        --group-name "$SECURITY_GROUP_NAME" \
```

```
        --description "ECS tutorial security group" \
        --vpc-id "$DEFAULT_VPC" \
        --tag-specifications "ResourceType=security-
group,Tags=[{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=$TUTORIAL_TAG}]"
 \
        --query 'GroupId' --output text)

    if [[ -z "$SECURITY_GROUP_ID" ]]; then
        log "ERROR: Failed to create security group"
        exit 1
    fi

    # Add HTTP access rule for nginx web server with restricted CIDR
    # SECURITY FIX: Restrict access to specific CIDR if available, otherwise
 document the risk
    log "WARNING: Security group allows HTTP (port 80) from 0.0.0.0/0 - restrict
 this in production"
    aws ec2 authorize-security-group-ingress \
        --group-id "$SECURITY_GROUP_ID" \
        --protocol tcp \
        --port 80 \
        --cidr "0.0.0.0/0"

    log "Created security group: $SECURITY_GROUP_ID"
    log "Added HTTP access on port 80"
    CREATED_RESOURCES+=("Security Group: $SECURITY_GROUP_ID")
}

# Function to get ECS optimized AMI
get_ecs_ami() {
    log "Getting ECS-optimized AMI ID..."

    ECS_AMI_ID=$(aws ssm get-parameters \
        --names /aws/service/ecs/optimized-ami/amazon-linux-2/recommended \
        --query 'Parameters[0].Value' --output text | jq -r '.image_id')

    if [[ -z "$ECS_AMI_ID" ]]; then
        log "ERROR: Failed to get ECS-optimized AMI ID"
        exit 1
    fi

    log "ECS-optimized AMI ID: $ECS_AMI_ID"
}
```

```
# Function to create IAM role for ECS instance (if it doesn't exist)
ensure_ecs_instance_role() {
    log "Checking for ecsInstanceRole..."

    if ! aws iam get-role --role-name ecsInstanceRole &> /dev/null; then
        log "Creating ecsInstanceRole..."

        # Create trust policy
        cat > ecs-instance-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

        # SECURITY FIX: Validate JSON before using
        if ! jq empty ecs-instance-trust-policy.json 2>/dev/null; then
            log "ERROR: Invalid JSON in trust policy"
            rm -f ecs-instance-trust-policy.json
            exit 1
        fi

        # Create role
        aws iam create-role \
            --role-name ecsInstanceRole \
            --assume-role-policy-document file://ecs-instance-trust-policy.json

        aws iam tag-role --role-name ecsInstanceRole --tags Key=project,Value=
$PROJECT_TAG Key=tutorial,Value=$TUTORIAL_TAG

        # Attach managed policy
        aws iam attach-role-policy \
            --role-name ecsInstanceRole \
            --policy-arn arn:aws:iam::aws:policy/service-role/
AmazonEC2ContainerServiceforEC2Role
```

```
        # Create instance profile
        aws iam create-instance-profile --instance-profile-name ecsInstanceRole

        # Add role to instance profile
        aws iam add-role-to-instance-profile \
            --instance-profile-name ecsInstanceRole \
            --role-name ecsInstanceRole

        # FIXED: Enhanced wait for role to be ready
        log "Waiting for IAM role to be ready..."
        aws iam wait role-exists --role-name ecsInstanceRole
        sleep 30  # Additional buffer for eventual consistency

        rm -f ecs-instance-trust-policy.json
        log "Created ecsInstanceRole"
        CREATED_RESOURCES+=("IAM Role: ecsInstanceRole")
    else
        log "ecsInstanceRole already exists"
    fi
}

# Function to launch container instance
launch_container_instance() {
    log "Launching ECS container instance..."

    # Create user data script
    cat > ecs-user-data.sh << EOF
#!/bin/bash
echo ECS_CLUSTER=$CLUSTER_NAME >> /etc/ecs/ecs.config
EOF

    # SECURITY FIX: Validate user data script before use
    if ! bash -n ecs-user-data.sh 2>/dev/null; then
        log "ERROR: Invalid user data script"
        rm -f ecs-user-data.sh
        exit 1
    fi

    INSTANCE_ID=$(aws ec2 run-instances \
        --image-id "$ECS_AMI_ID" \
        --instance-type t3.micro \
        --key-name "$KEY_PAIR_NAME" \
        --security-group-ids "$SECURITY_GROUP_ID" \
        --subnet-id "$DEFAULT_SUBNET" \
```

```
        --iam-instance-profile Name=ecsInstanceRole \
        --user-data file://ecs-user-data.sh \
        --tag-specifications "ResourceType=instance,Tags=[{Key=Name,Value=ecs-
tutorial-instance},{Key=project,Value=$PROJECT_TAG},{Key=tutorial,Value=
$TUTORIAL_TAG}]" \
        --monitoring Enabled=false \
        --metadata-options HttpTokens=required,HttpPutResponseHopLimit=1 \
        --query 'Instances[0].InstanceId' --output text)

    if [[ -z "$INSTANCE_ID" ]]; then
        log "ERROR: Failed to launch EC2 instance"
        rm -f ecs-user-data.sh
        exit 1
    fi

    log "Launched EC2 instance: $INSTANCE_ID"
    log "Instance metadata options: IMDSv2 enforced with hop limit 1"
    CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

    # Wait for instance to be running
    log "Waiting for instance to be running..."
    aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

    # Wait for ECS agent to register
    log "Waiting for ECS agent to register with cluster..."
    local max_attempts=30
    local attempt=0

    while [[ $attempt -lt $max_attempts ]]; do
        CONTAINER_INSTANCES=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns' --output text)
        if [[ -n "$CONTAINER_INSTANCES" && "$CONTAINER_INSTANCES" != "None" ]];
 then
            log "Container instance registered successfully"
            break
        fi

        attempt=$((attempt + 1))
        log "Waiting for container instance registration... (attempt $attempt/
$max_attempts)"
        sleep 10
    done

    if [[ $attempt -eq $max_attempts ]]; then
```

```
        log "ERROR: Container instance failed to register within expected time"
        rm -f ecs-user-data.sh
        exit 1
    fi

    rm -f ecs-user-data.sh
}

# Function to register task definition
register_task_definition() {
    log "Creating task definition..."

    # Create nginx task definition JSON matching the tutorial
    cat > task-definition.json << 'EOF'
{
    "family": "TASK_FAMILY_PLACEHOLDER",
    "containerDefinitions": [
        {
            "name": "nginx",
            "image": "public.ecr.aws/docker/library/nginx:latest",
            "cpu": 256,
            "memory": 512,
            "essential": true,
            "portMappings": [
                {
                    "containerPort": 80,
                    "hostPort": 80,
                    "protocol": "tcp"
                }
            ],
            "logConfiguration": {
                "logDriver": "awslogs",
                "options": {
                    "awslogs-group": "/ecs/nginx-task",
                    "awslogs-region": "REGION_PLACEHOLDER",
                    "awslogs-stream-prefix": "ecs"
                }
            }
        }
    ],
    "requiresCompatibilities": ["EC2"],
    "networkMode": "bridge",
    "tags": [
        {
```

```
            "key": "project",
            "value": "PROJECT_TAG_PLACEHOLDER"
        },
        {
            "key": "tutorial",
            "value": "TUTORIAL_TAG_PLACEHOLDER"
        }
    ]
}
EOF

    # Replace placeholders securely
    sed -i "s|TASK_FAMILY_PLACEHOLDER|$TASK_FAMILY|g" task-definition.json
    sed -i "s|REGION_PLACEHOLDER|$AWS_REGION|g" task-definition.json
    sed -i "s|PROJECT_TAG_PLACEHOLDER|$PROJECT_TAG|g" task-definition.json
    sed -i "s|TUTORIAL_TAG_PLACEHOLDER|$TUTORIAL_TAG|g" task-definition.json

    # FIXED: Validate JSON before registration
    if ! jq empty task-definition.json 2>/dev/null; then
        log "ERROR: Invalid JSON in task definition"
        rm -f task-definition.json
        exit 1
    fi

    TASK_DEFINITION_ARN=$(aws ecs register-task-definition \
        --cli-input-json file://task-definition.json \
        --query 'taskDefinition.taskDefinitionArn' --output text)

    if [[ -z "$TASK_DEFINITION_ARN" ]]; then
        log "ERROR: Failed to register task definition"
        rm -f task-definition.json
        exit 1
    fi

    log "Registered task definition: $TASK_DEFINITION_ARN"
    log "Task definition includes CloudWatch Logs configuration for monitoring"
    CREATED_RESOURCES+=("Task Definition: $TASK_DEFINITION_ARN")

    rm -f task-definition.json
}

# Function to create service
create_service() {
    log "Creating ECS service..."
```

```
    SERVICE_ARN=$(aws ecs create-service \
        --cluster "$CLUSTER_NAME" \
        --service-name "$SERVICE_NAME" \
        --task-definition "$TASK_FAMILY" \
        --desired-count 1 \
        --tags key=project,value=$PROJECT_TAG key=tutorial,value=$TUTORIAL_TAG \
        --query 'service.serviceArn' --output text)

    if [[ -z "$SERVICE_ARN" ]]; then
        log "ERROR: Failed to create service"
        exit 1
    fi

    log "Created service: $SERVICE_ARN"
    CREATED_RESOURCES+=("ECS Service: $SERVICE_NAME")

    # Wait for service to be stable
    log "Waiting for service to be stable..."
    aws ecs wait services-stable --cluster "$CLUSTER_NAME" --services
 "$SERVICE_NAME"

    log "Service is now stable and running"

    # Get the task ARN for monitoring
    TASK_ARN=$(aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name
 "$SERVICE_NAME" --query 'taskArns[0]' --output text)
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Service task: $TASK_ARN"
        CREATED_RESOURCES+=("ECS Task: $TASK_ARN")
    fi
}

# Function to demonstrate monitoring and testing
demonstrate_monitoring() {
    log "Demonstrating monitoring capabilities..."

    # List services
    log "Listing services in cluster:"
    aws ecs list-services --cluster "$CLUSTER_NAME" --output table

    # Describe service
    log "Service details:"
```

```
    aws ecs describe-services --cluster "$CLUSTER_NAME" --
services "$SERVICE_NAME" --output table --query 'services[0].
{ServiceName:serviceName,Status:status,RunningCount:runningCount,DesiredCount:desiredCoun

    # List tasks
    log "Listing tasks in service:"
    aws ecs list-tasks --cluster "$CLUSTER_NAME" --service-name "$SERVICE_NAME"
 --output table

    # Describe task
    if [[ -n "$TASK_ARN" && "$TASK_ARN" != "None" ]]; then
        log "Task details:"
        aws ecs describe-tasks --cluster "$CLUSTER_NAME"
 --tasks "$TASK_ARN" --output table --query 'tasks[0].
{TaskArn:taskArn,LastStatus:lastStatus,DesiredStatus:desiredStatus,CreatedAt:createdAt}'
    fi

    # List container instances
    log "Container instances in cluster:"
    aws ecs list-container-instances --cluster "$CLUSTER_NAME" --output table

    # Describe container instance
    CONTAINER_INSTANCE_ARN=$(aws ecs list-container-instances --cluster
 "$CLUSTER_NAME" --query 'containerInstanceArns[0]' --output text)
    if [[ -n "$CONTAINER_INSTANCE_ARN" && "$CONTAINER_INSTANCE_ARN" != "None" ]];
 then
        log "Container instance details:"
        aws ecs describe-container-instances --cluster "$CLUSTER_NAME"
 --container-instances "$CONTAINER_INSTANCE_ARN" --output table --query
 'containerInstances[0].
{Arn:containerInstanceArn,Status:status,RunningTasks:runningTasksCount,PendingTasks:pendi
    fi

    # Test the nginx web server
    log "Testing nginx web server..."
    PUBLIC_IP=$(aws ec2 describe-instances --instance-ids "$INSTANCE_ID" --query
 'Reservations[0].Instances[0].PublicIpAddress' --output text)

    if [[ -n "$PUBLIC_IP" && "$PUBLIC_IP" != "None" ]]; then
        log "Container instance public IP: $PUBLIC_IP"
        log "Testing HTTP connection to nginx..."

        # Wait a moment for nginx to be fully ready
        sleep 10
```

```
        if curl -s --connect-timeout 10 "http://$PUBLIC_IP" | grep -q "Welcome to
 nginx"; then
            log "SUCCESS: Nginx web server is responding correctly"
            echo ""
            echo "============================================="
            echo "WEB SERVER TEST SUCCESSFUL"
            echo "============================================="
            echo "You can access your nginx web server at: http://$PUBLIC_IP"
            echo "The nginx welcome page should be visible in your browser."
        else
            log "WARNING: Nginx web server may not be fully ready yet. Try
 accessing http://$PUBLIC_IP in a few minutes."
        fi
    else
        log "WARNING: Could not retrieve public IP address"
    fi
}

# Main execution
main() {
    log "Starting ECS EC2 Launch Type Tutorial (UPDATED VERSION)"
    log "Log file: $LOG_FILE"
    log "Security improvements: IMDSv2 enforced, JSON validation, input
 sanitization, CloudWatch Logs configured"

    check_prerequisites
    create_cluster
    create_key_pair
    create_security_group
    get_ecs_ami
    ensure_ecs_instance_role
    launch_container_instance
    register_task_definition
    create_service
    demonstrate_monitoring

    log "Tutorial completed successfully!"

    echo ""
    echo "============================================="
    echo "TUTORIAL COMPLETED SUCCESSFULLY"
    echo "============================================="
    echo "Resources created:"
```

```
    for resource in "${CREATED_RESOURCES[@]}"; do
        echo "  - $resource"
    done
    echo ""
    echo "The nginx service will continue running and maintain the desired task
 count."
    echo "You can monitor the service status using:"
    echo "  aws ecs describe-services --cluster $CLUSTER_NAME --services
 $SERVICE_NAME"
    echo ""
    if [[ -n "${PUBLIC_IP:-}" ]]; then
        echo "Access your web server at: http://$PUBLIC_IP"
        echo ""
    fi
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    log "Auto-confirming cleanup - proceeding with resource cleanup"
    cleanup_resources
    log "All resources have been cleaned up"
}

# Run main function
main "$@"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - AuthorizeSecurityGroupIngress
  - CreateCluster
  - CreateInstanceProfile
  - CreateKeyPair
  - CreateRole
  - CreateSecurityGroup
  - CreateService
  - DeleteCluster
  - DeleteKeyPair
  - DeleteSecurityGroup

- DeleteService
- DeregisterTaskDefinition
- DescribeContainerInstances
- DescribeInstances
- DescribeServices
- DescribeSubnets
- DescribeTasks
- DescribeVpcs
- GetCallerIdentity
- GetParameters
- GetRole
- ListContainerInstances
- ListServices
- ListTasks
- RegisterTaskDefinition
- RunInstances
- StopTask
- TerminateInstances
- UpdateService
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

创建 Amazon Managed Grafana 工作区

以下代码示例展示了如何：

- 为工作区创建 IAM 角色
- 创建 Grafana 工作区
- 配置身份验证
- 配置可选设置

- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon Managed Grafana Workspace Creation Script
# This script creates an Amazon Managed Grafana workspace and configures it

# Set up logging
LOG_FILE="grafana-workspace-creation.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon Managed Grafana workspace creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error\|exception\|fail" > /dev/null; then
        echo "ERROR: Command '$cmd' failed with output:"
        echo "$output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..."

    if [ -n "$WORKSPACE_ID" ]; then
```

```
        echo "Deleting workspace $WORKSPACE_ID..."
        aws grafana delete-workspace --workspace-id "$WORKSPACE_ID"
    fi

    if [ -n "$ROLE_NAME" ]; then
        echo "Detaching policies from role $ROLE_NAME..."
        if [ -n "$POLICY_ARN" ]; then
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$POLICY_ARN"
        fi

        echo "Deleting role $ROLE_NAME..."
        aws iam delete-role --role-name "$ROLE_NAME"
    fi

    if [ -n "$POLICY_ARN" ]; then
        echo "Deleting policy..."
        aws iam delete-policy --policy-arn "$POLICY_ARN"
    fi

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed. See $LOG_FILE for details."
}

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
WORKSPACE_NAME="GrafanaWorkspace-${RANDOM_ID}"
ROLE_NAME="GrafanaWorkspaceRole-${RANDOM_ID}"

echo "Using workspace name: $WORKSPACE_NAME"
echo "Using role name: $ROLE_NAME"

# Step 1: Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "AWS Account ID: $ACCOUNT_ID"

# Step 2: Create IAM role for Grafana workspace
echo "Creating IAM role for Grafana workspace..."

# Create trust policy document
```

```
cat > trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "grafana.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role
ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$ROLE_NAME" \
  --assume-role-policy-document file://trust-policy.json \
  --description "Role for Amazon Managed Grafana workspace")

check_error "$ROLE_OUTPUT" "create-role"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-grafana-gs
echo "IAM role created successfully"

# Extract role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "Role ARN: $ROLE_ARN"

# Attach policies to the role
echo "Attaching policies to the role..."

# CloudWatch policy
cat > cloudwatch-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cloudwatch:DescribeAlarmsForMetric",
        "cloudwatch:DescribeAlarmHistory",
        "cloudwatch:DescribeAlarms",
```

```
        "cloudwatch:ListMetrics",
        "cloudwatch:GetMetricStatistics",
        "cloudwatch:GetMetricData"
      ],
      "Resource": "*"
    }
  ]
}
EOF

POLICY_OUTPUT=$(aws iam create-policy \
  --policy-name "GrafanaCloudWatchPolicy-${RANDOM_ID}" \
  --policy-document file://cloudwatch-policy.json)

check_error "$POLICY_OUTPUT" "create-policy"

POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)
echo "CloudWatch policy ARN: $POLICY_ARN"

ATTACH_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$ROLE_NAME" \
  --policy-arn "$POLICY_ARN")

check_error "$ATTACH_OUTPUT" "attach-role-policy"
echo "CloudWatch policy attached to role"

# Step 3: Create the Grafana workspace
echo "Creating Amazon Managed Grafana workspace..."
WORKSPACE_OUTPUT=$(aws grafana create-workspace \
  --workspace-name "$WORKSPACE_NAME" \
  --authentication-providers "SAML" \
  --permission-type "CUSTOMER_MANAGED" \
  --account-access-type "CURRENT_ACCOUNT" \
  --workspace-role-arn "$ROLE_ARN" \
  --workspace-data-sources "CLOUDWATCH" "PROMETHEUS" "XRAY" \
  --grafana-version "10.4" \
  --tags Environment=Development,project=doc-smith,tutorial=amazon-managed-
grafana-gs)

check_error "$WORKSPACE_OUTPUT" "create-workspace"

echo "Workspace creation initiated:"
echo "$WORKSPACE_OUTPUT"
```

```
# Extract workspace ID
WORKSPACE_ID=$(echo "$WORKSPACE_OUTPUT" | grep -o '"id": "[^"]*' | cut -d'"' -f4)

if [ -z "$WORKSPACE_ID" ]; then
    echo "ERROR: Failed to extract workspace ID from output"
    exit 1
fi

echo "Workspace ID: $WORKSPACE_ID"

# Step 4: Wait for workspace to become active
echo "Waiting for workspace to become active. This may take several minutes..."
ACTIVE=false
MAX_ATTEMPTS=30
ATTEMPT=0

while [ $ACTIVE = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
    ATTEMPT=$((ATTEMPT+1))
    echo "Checking workspace status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

    DESCRIBE_OUTPUT=$(aws grafana describe-workspace --workspace-id
 "$WORKSPACE_ID")
    check_error "$DESCRIBE_OUTPUT" "describe-workspace"

    STATUS=$(echo "$DESCRIBE_OUTPUT" | grep -o '"status": "[^"]*' | cut -d'"' -
f4)
    echo "Current status: $STATUS"

    if [ "$STATUS" = "ACTIVE" ]; then
        ACTIVE=true
        echo "Workspace is now ACTIVE"
    elif [ "$STATUS" = "FAILED" ]; then
        echo "ERROR: Workspace creation failed"
        cleanup_on_error
        exit 1
    else
        echo "Workspace is still being created. Waiting 30 seconds..."
        sleep 30
    fi
done

if [ $ACTIVE = false ]; then
    echo "ERROR: Workspace did not become active within the expected time"
    cleanup_on_error
```

```
    exit 1
fi

# Extract workspace endpoint URL
WORKSPACE_URL=$(echo "$DESCRIBE_OUTPUT" | grep -o '"endpoint": "[^"]*' | cut -
d'"' -f4)
echo "Workspace URL: https://$WORKSPACE_URL"

# Step 5: Display workspace information
echo ""
echo "==========================================="
echo "WORKSPACE INFORMATION"
echo "==========================================="
echo "Workspace ID: $WORKSPACE_ID"
echo "Workspace URL: https://$WORKSPACE_URL"
echo "Workspace Name: $WORKSPACE_NAME"
echo "IAM Role: $ROLE_NAME"
echo ""
echo "Note: Since SAML authentication is used, you need to configure SAML
 settings"
echo "using the AWS Management Console or the update-workspace-authentication
 command."
echo "==========================================="

# Step 6: Prompt for cleanup
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Resources created:"
echo "- Amazon Managed Grafana workspace: $WORKSPACE_ID"
echo "- IAM Role: $ROLE_NAME"
echo "- IAM Policy: GrafanaCloudWatchPolicy-${RANDOM_ID}"
echo ""
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy] ]]; then
    echo "Cleaning up resources..."

    echo "Deleting workspace $WORKSPACE_ID..."
    DELETE_OUTPUT=$(aws grafana delete-workspace --workspace-id "$WORKSPACE_ID")
    check_error "$DELETE_OUTPUT" "delete-workspace"
```

```
    echo "Waiting for workspace to be deleted..."
    DELETED=false
    ATTEMPT=0

    while [ $DELETED = false ] && [ $ATTEMPT -lt $MAX_ATTEMPTS ]; do
        ATTEMPT=$((ATTEMPT+1))
        echo "Checking deletion status (attempt $ATTEMPT of $MAX_ATTEMPTS)..."

        if aws grafana describe-workspace --workspace-id "$WORKSPACE_ID" 2>&1 |
 grep -i "not found\|does not exist" > /dev/null; then
            DELETED=true
            echo "Workspace has been deleted"
        else
            echo "Workspace is still being deleted. Waiting 30 seconds..."
            sleep 30
        fi
    done

    if [ $DELETED = false ]; then
        echo "WARNING: Workspace deletion is taking longer than expected. It may
 still be in progress."
    fi

    # Detach policy from role
    echo "Detaching policy from role..."
    aws iam detach-role-policy \
      --role-name "$ROLE_NAME" \
      --policy-arn "$POLICY_ARN"

    # Delete policy
    echo "Deleting IAM policy..."
    aws iam delete-policy \
      --policy-arn "$POLICY_ARN"

    # Delete role
    echo "Deleting IAM role..."
    aws iam delete-role \
      --role-name "$ROLE_NAME"

    # Clean up JSON files
    rm -f trust-policy.json cloudwatch-policy.json

    echo "Cleanup completed"
else
```

```
    echo "Skipping cleanup. Resources will remain in your AWS account."
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreatePolicy
  - CreateRole
  - CreateWorkspace
  - DeletePolicy
  - DeleteRole
  - DeleteWorkspace
  - DescribeWorkspace
  - DetachRolePolicy
  - GetCallerIdentity

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon 开发工具包获取需要具有 Amazon STS 的 MFA 令牌的会话令牌

以下代码示例显示如何获取需要 MFA 令牌的会话令牌。

> **Warning**
>
> 为了避免安全风险，在开发专用软件或处理真实数据时，请勿使用 IAM 用户进行身份验证。而是使用与身份提供商的联合身份验证，例如 Amazon IAM Identity Center。

- 创建一个 IAM 角色，该角色授予列出 Amazon S3 存储桶的权限。
- 创建一个 IAM 用户，该用户仅在提供 MFA 凭证时才有权代入角色。
- 为该用户注册一个 MFA 设备。
- 提供 MFA 凭证以获取会话令牌，并使用临时凭证列出 S3 存储桶。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 查看 GitHub，了解更多信息。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个 IAM 用户，注册一个 MFA 设备，然后创建一个角色，该角色授予仅在使用 MFA 凭证时允许用户列出 S3 存储桶的权限。

```
def setup(iam_resource):
    """
    Creates a new user with no permissions.
    Creates a new virtual multi-factor authentication (MFA) device.
    Displays the QR code to seed the device.
    Asks for two codes from the MFA device.
    Registers the MFA device for the user.
    Creates an access key pair for the user.
    Creates an inline policy for the user that lets the user list Amazon S3
 buckets,
    but only when MFA credentials are used.

    Any MFA device that can scan a QR code will work with this demonstration.
    Common choices are mobile apps like LastPass Authenticator,
    Microsoft Authenticator, or Google Authenticator.

    :param iam_resource: A Boto3 AWS Identity and Access Management (IAM)
 resource
                         that has permissions to create users, MFA devices, and
                         policies in the account.
    :return: The newly created user, user key, and virtual MFA device.
    """
    user = iam_resource.create_user(UserName=unique_name("user"))
    print(f"Created user {user.name}.")

    virtual_mfa_device = iam_resource.create_virtual_mfa_device(
        VirtualMFADeviceName=unique_name("mfa")
    )
    print(f"Created virtual MFA device {virtual_mfa_device.serial_number}")
```

```
    print(
        f"Showing the QR code for the device. Scan this in the MFA app of your "
        f"choice."
    )
    with open("qr.png", "wb") as qr_file:
        qr_file.write(virtual_mfa_device.qr_code_png)
    webbrowser.open(qr_file.name)

    print(f"Enter two consecutive code from your MFA device.")
    mfa_code_1 = input("Enter the first code: ")
    mfa_code_2 = input("Enter the second code: ")
    user.enable_mfa(
        SerialNumber=virtual_mfa_device.serial_number,
        AuthenticationCode1=mfa_code_1,
        AuthenticationCode2=mfa_code_2,
    )
    os.remove(qr_file.name)
    print(f"MFA device is registered with the user.")

    user_key = user.create_access_key_pair()
    print(f"Created access key pair for user.")

    print(f"Wait for user to be ready.", end="")
    progress_bar(10)

    user.create_policy(
        PolicyName=unique_name("user-policy"),
        PolicyDocument=json.dumps(
            {
                "Version":"2012-10-17",
                "Statement": [
                    {
                        "Effect": "Allow",
                        "Action": "s3:ListAllMyBuckets",
                        "Resource": "arn:aws:s3:::*",
                        "Condition": {"Bool": {"aws:MultiFactorAuthPresent":
True}},
                    }
                ],
            }
        ),
    )
    print(
```

```
        f"Created an inline policy for {user.name} that lets the user list
 buckets, "
        f"but only when MFA credentials are present."
    )

    print("Give AWS time to propagate these new resources and connections.",
 end="")
    progress_bar(10)

    return user, user_key, virtual_mfa_device
```

通过传递 MFA 令牌获取临时会话凭证，并使用凭证列出账户的 S3 存储桶。

```
def list_buckets_with_session_token_with_mfa(mfa_serial_number, mfa_totp,
 sts_client):
    """
    Gets a session token with MFA credentials and uses the temporary session
    credentials to list Amazon S3 buckets.

    Requires an MFA device serial number and token.

    :param mfa_serial_number: The serial number of the MFA device. For a virtual
 MFA
                              device, this is an Amazon Resource Name (ARN).
    :param mfa_totp: A time-based, one-time password issued by the MFA device.
    :param sts_client: A Boto3 STS instance that has permission to assume the
 role.
    """
    if mfa_serial_number is not None:
        response = sts_client.get_session_token(
            SerialNumber=mfa_serial_number, TokenCode=mfa_totp
        )
    else:
        response = sts_client.get_session_token()
    temp_credentials = response["Credentials"]

    s3_resource = boto3.resource(
        "s3",
        aws_access_key_id=temp_credentials["AccessKeyId"],
        aws_secret_access_key=temp_credentials["SecretAccessKey"],
```

```
        aws_session_token=temp_credentials["SessionToken"],
    )

    print(f"Buckets for the account:")
    for bucket in s3_resource.buckets.all():
        print(bucket.name)
```

销毁为演示创建的资源。

```
def teardown(user, virtual_mfa_device):
    """
    Removes all resources created during setup.

    :param user: The demo user.
    :param role: The demo MFA device.
    """
    for user_pol in user.policies.all():
        user_pol.delete()
        print("Deleted inline user policy.")
    for key in user.access_keys.all():
        key.delete()
        print("Deleted user's access key.")
    for mfa in user.mfa_devices.all():
        mfa.disassociate()
    virtual_mfa_device.delete()
    user.delete()
    print(f"Deleted {user.name}.")
```

使用之前定义的函数运行此方案。

```
def usage_demo():
    """Drives the demonstration."""
    print("-" * 88)
    print(
        f"Welcome to the AWS Security Token Service assume role demo, "
        f"starring multi-factor authentication (MFA)!"
    )
```

```
    print("-" * 88)
    iam_resource = boto3.resource("iam")
    user, user_key, virtual_mfa_device = setup(iam_resource)
    try:
        sts_client = boto3.client(
            "sts", aws_access_key_id=user_key.id,
aws_secret_access_key=user_key.secret
        )
        try:
            print("Listing buckets without specifying MFA credentials.")
            list_buckets_with_session_token_with_mfa(None, None, sts_client)
        except ClientError as error:
            if error.response["Error"]["Code"] == "AccessDenied":
                print("Got expected AccessDenied error.")
        mfa_totp = input("Enter the code from your registered MFA device: ")
        list_buckets_with_session_token_with_mfa(
            virtual_mfa_device.serial_number, mfa_totp, sts_client
        )
    finally:
        teardown(user, virtual_mfa_device)
        print("Thanks for watching!")
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的 GetSessionToken。

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用 Amazon ECR

以下代码示例展示了如何：

- 创建 Docker 映像
- 创建 Amazon ECR 存储库
- 删除资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon ECR Getting Started Script
# This script demonstrates the lifecycle of a Docker image in Amazon ECR

# Set up logging
LOG_FILE="ecr-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "================================================="
echo "Amazon ECR Getting Started Tutorial"
echo "================================================="
echo "This script will:"
echo "1. Create a Docker image"
echo "2. Create an Amazon ECR repository"
echo "3. Authenticate to Amazon ECR"
echo "4. Push the image to Amazon ECR"
echo "5. Pull the image from Amazon ECR"
echo "6. Clean up resources (optional)"
echo "================================================="

# Check prerequisites
echo "Checking prerequisites..."

# Check if AWS CLI is installed
if ! command -v aws &> /dev/null; then
    echo "ERROR: AWS CLI is not installed. Please install it before running this
 script."
    echo "Visit https://docs.aws.amazon.com/cli/latest/userguide/install-
cliv2.html for installation instructions."
    exit 1
fi
```

```
# Check if AWS CLI is configured
if ! aws sts get-caller-identity &> /dev/null; then
    echo "ERROR: AWS CLI is not configured properly. Please run 'aws configure'
 to set up your credentials."
    exit 1
fi

# Check if Docker is installed
if ! command -v docker &> /dev/null; then
    echo "ERROR: Docker is not installed. Please install Docker before running
 this script."
    echo "Visit https://docs.docker.com/get-docker/ for installation
 instructions."
    exit 1
fi

# Check if Docker daemon is running
if ! docker info &> /dev/null; then
    echo "ERROR: Docker daemon is not running. Please start Docker and try
 again."
    exit 1
fi

echo "All prerequisites met."

# Initialize variables
REPO_URI=""
TIMEOUT_CMD="timeout 300"  # 5-minute timeout for long-running commands

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Check the log file for details: $LOG_FILE"

    echo "================================================="
    echo "Resources created:"
    echo "- Docker image: hello-world (local)"
    if [ -n "$REPO_URI" ]; then
        echo "- ECR Repository: hello-repository"
        echo "- ECR Image: $REPO_URI:latest"
    fi
    echo "================================================="
```

```
    echo "Attempting to clean up resources..."
    cleanup
    exit 1
}

# Function to clean up resources
cleanup() {
    echo "================================================="
    echo "Cleaning up resources..."

    # Delete the image from ECR if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting image from ECR repository..."
        aws ecr batch-delete-image --repository-name hello-repository --image-ids
 imageTag=latest || echo "Failed to delete image, it may not exist or may have
 already been deleted."
    fi

    # Delete the ECR repository if it exists
    if [ -n "$REPO_URI" ]; then
        echo "Deleting ECR repository..."
        aws ecr delete-repository --repository-name hello-repository --force ||
 echo "Failed to delete repository, it may not exist or may have already been
 deleted."
    fi

    # Remove local Docker image
    echo "Removing local Docker image..."
    docker rmi hello-world:latest 2>/dev/null || echo "Failed to remove local
 image, it may not exist or may have already been deleted."
    if [ -n "$REPO_URI" ]; then
        docker rmi "$REPO_URI:latest" 2>/dev/null || echo "Failed to remove
 tagged image, it may not exist or may have already been deleted."
    fi

    echo "Cleanup completed."
    echo "================================================="
}

# Step 1: Create a Docker image
echo "Step 1: Creating a Docker image"

# Create Dockerfile
echo "Creating Dockerfile..."
```

```
cat > Dockerfile << 'EOF'
FROM public.ecr.aws/amazonlinux/amazonlinux:latest

# Install dependencies
RUN yum update -y && \
 yum install -y httpd

# Install apache and write hello world message
RUN echo 'Hello World!' > /var/www/html/index.html

# Configure apache
RUN echo 'mkdir -p /var/run/httpd' >> /root/run_apache.sh && \
 echo 'mkdir -p /var/lock/httpd' >> /root/run_apache.sh && \
 echo '/usr/sbin/httpd -D FOREGROUND' >> /root/run_apache.sh && \
 chmod 755 /root/run_apache.sh

EXPOSE 80

CMD /root/run_apache.sh
EOF

# Build Docker image
echo "Building Docker image..."
$TIMEOUT_CMD docker build -t hello-world . || handle_error "Failed to build
 Docker image or operation timed out after 5 minutes"

# Verify image was created
echo "Verifying Docker image..."
docker images --filter reference=hello-world || handle_error "Failed to list
 Docker images"

echo "Docker image created successfully."

# Step 2: Create an Amazon ECR repository
echo "Step 2: Creating an Amazon ECR repository"

# Get AWS account ID
echo "Getting AWS account ID..."
AWS_ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ -z "$AWS_ACCOUNT_ID" || "$AWS_ACCOUNT_ID" == *"error"* ]]; then
    handle_error "Failed to get AWS account ID. Make sure your AWS credentials
 are configured correctly."
fi
echo "AWS Account ID: $AWS_ACCOUNT_ID"
```

```
# Get current region
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
    echo "No AWS region configured, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Create ECR repository
echo "Creating ECR repository..."
REPO_RESULT=$(aws ecr create-repository --repository-name hello-repository --
tags key=project,value=doc-smith key=tutorial,value=amazon-elastic-container-
registry-gs)
if [[ -z "$REPO_RESULT" || "$REPO_RESULT" == *"error"* ]]; then
    handle_error "Failed to create ECR repository"
fi

# Extract repository URI
REPO_URI="$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com/hello-repository"
echo "Repository URI: $REPO_URI"

# Step 3: Authenticate to Amazon ECR
echo "Step 3: Authenticating to Amazon ECR"

echo "Getting ECR login password..."
aws ecr get-login-password --region "$AWS_REGION" | docker login --username
 AWS --password-stdin "$AWS_ACCOUNT_ID.dkr.ecr.$AWS_REGION.amazonaws.com" ||
 handle_error "Failed to authenticate to ECR"

echo "Successfully authenticated to ECR."

# Step 4: Push the image to Amazon ECR
echo "Step 4: Pushing the image to Amazon ECR"

# Tag the image
echo "Tagging Docker image..."
docker tag hello-world:latest "$REPO_URI:latest" || handle_error "Failed to tag
 Docker image"

# Push the image with timeout
echo "Pushing image to ECR..."
```

```
$TIMEOUT_CMD docker push "$REPO_URI:latest" || handle_error "Failed to push image
 to ECR or operation timed out after 5 minutes"

echo "Successfully pushed image to ECR."

# Step 5: Pull the image from Amazon ECR
echo "Step 5: Pulling the image from Amazon ECR"

# Remove local image to ensure we're pulling from ECR
echo "Removing local tagged image..."
docker rmi "$REPO_URI:latest" || echo "Warning: Failed to remove local tagged
 image"

# Pull the image with timeout
echo "Pulling image from ECR..."
$TIMEOUT_CMD docker pull "$REPO_URI:latest" || handle_error "Failed to pull image
 from ECR or operation timed out after 5 minutes"

echo "Successfully pulled image from ECR."

# List resources created
echo "================================================="
echo "Resources created:"
echo "- Docker image: hello-world (local)"
echo "- ECR Repository: hello-repository"
echo "- ECR Image: $REPO_URI:latest"
echo "================================================="

# Ask user if they want to clean up resources
echo ""
echo "========================================="
echo "CLEANUP CONFIRMATION"
echo "========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    # Step 6: Delete the image from ECR
    echo "Step 6: Deleting the image from ECR"

    DELETE_IMAGE_RESULT=$(aws ecr batch-delete-image --repository-name hello-
repository --image-ids imageTag=latest)
    if [[ -z "$DELETE_IMAGE_RESULT" || "$DELETE_IMAGE_RESULT" == *"error"* ]];
 then
```

```
        echo "Warning: Failed to delete image from ECR"
    else
        echo "Successfully deleted image from ECR."
    fi

    # Step 7: Delete the ECR repository
    echo "Step 7: Deleting the ECR repository"

    DELETE_REPO_RESULT=$(aws ecr delete-repository --repository-name hello-
repository --force)
    if [[ -z "$DELETE_REPO_RESULT" || "$DELETE_REPO_RESULT" == *"error"* ]]; then
        echo "Warning: Failed to delete ECR repository"
    else
        echo "Successfully deleted ECR repository."
    fi

    # Remove local Docker images
    echo "Removing local Docker images..."
    docker rmi hello-world:latest 2>/dev/null || echo "Warning: Failed to remove
 local image"

    echo "All resources have been cleaned up."
else
    echo "Resources were not cleaned up. You can manually clean up later with:"
    echo "aws ecr batch-delete-image --repository-name hello-repository --image-
ids imageTag=latest"
    echo "aws ecr delete-repository --repository-name hello-repository --force"
    echo "docker rmi hello-world:latest"
    echo "docker rmi $REPO_URI:latest"
fi

echo "================================================="
echo "Tutorial completed!"
echo "Log file: $LOG_FILE"
echo "================================================="
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - BatchDeleteImage
  - CreateRepository
  - DeleteRepository
  - GetCallerIdentity

- GetLoginPassword

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用 Amazon EKS

以下代码示例展示了如何：

- 为 EKS 集群创建 VPC
- 为 EKS 集群创建 IAM 角色
- 创建您的 EKS 集群
- 配置 kubectl 以与集群通信
- 创建托管节点组
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon EKS Cluster Creation Script (v2)
# This script creates an Amazon EKS cluster with a managed node group using the
 AWS CLI

# Set up logging
LOG_FILE="eks-cluster-creation-v2.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon EKS cluster creation script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command success
check_command() {
    if [ $? -ne 0 ] || echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to check if kubectl is installed
check_kubectl() {
    if ! command -v kubectl &> /dev/null; then
        echo "WARNING: kubectl is not installed or not in your PATH."
        echo ""
        echo "To install kubectl, follow these instructions based on your
 operating system:"
        echo ""
        echo "For Linux:"
        echo "  1. Download the latest release:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl\""
        echo ""
        echo "  2. Make the kubectl binary executable:"
        echo "     chmod +x ./kubectl"
        echo ""
        echo "  3. Move the binary to your PATH:"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
        echo ""
        echo "For macOS:"
        echo "  1. Using Homebrew:"
        echo "     brew install kubectl"
        echo "     or"
        echo "  2. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/\$(curl -L -s https://
dl.k8s.io/release/stable.txt)/bin/darwin/amd64/kubectl\""
        echo "     chmod +x ./kubectl"
        echo "     sudo mv ./kubectl /usr/local/bin/kubectl"
```

```
        echo ""
        echo "For Windows:"
        echo "  1. Using curl:"
        echo "     curl -LO \"https://dl.k8s.io/release/v1.28.0/bin/windows/
amd64/kubectl.exe\""
        echo "     Add the binary to your PATH"
        echo "     or"
        echo "  2. Using Chocolatey:"
        echo "     choco install kubernetes-cli"
        echo ""
        echo "After installation, verify with: kubectl version --client"
        echo ""
        return 1
    fi
    return 0
}

# Generate a random identifier for resource names
RANDOM_ID=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 6 | head -n 1)
STACK_NAME="eks-vpc-stack-${RANDOM_ID}"
CLUSTER_NAME="eks-cluster-${RANDOM_ID}"
NODEGROUP_NAME="eks-nodegroup-${RANDOM_ID}"
CLUSTER_ROLE_NAME="EKSClusterRole-${RANDOM_ID}"
NODE_ROLE_NAME="EKSNodeRole-${RANDOM_ID}"

echo "Using the following resource names:"
echo "- VPC Stack: $STACK_NAME"
echo "- EKS Cluster: $CLUSTER_NAME"
echo "- Node Group: $NODEGROUP_NAME"
echo "- Cluster IAM Role: $CLUSTER_ROLE_NAME"
echo "- Node IAM Role: $NODE_ROLE_NAME"

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources in reverse order..."

    # Check if node group exists and delete it
    if aws eks list-nodegroups --cluster-name "$CLUSTER_NAME" --query
 "nodegroups[?contains(@,'$NODEGROUP_NAME')]" --output text 2>/dev/null | grep -q
 "$NODEGROUP_NAME"; then
        echo "Deleting node group: $NODEGROUP_NAME"
```

```
        aws eks delete-nodegroup --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
        echo "Waiting for node group deletion to complete..."
        aws eks wait nodegroup-deleted --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME"
        echo "Node group deleted successfully."
    fi

    # Check if cluster exists and delete it
    if aws eks describe-cluster --name "$CLUSTER_NAME" 2>/dev/null; then
        echo "Deleting cluster: $CLUSTER_NAME"
        aws eks delete-cluster --name "$CLUSTER_NAME"
        echo "Waiting for cluster deletion to complete (this may take several
 minutes)..."
        aws eks wait cluster-deleted --name "$CLUSTER_NAME"
        echo "Cluster deleted successfully."
    fi

    # Check if CloudFormation stack exists and delete it
    if aws cloudformation describe-stacks --stack-name "$STACK_NAME" 2>/dev/null;
 then
        echo "Deleting CloudFormation stack: $STACK_NAME"
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "Waiting for CloudFormation stack deletion to complete..."
        aws cloudformation wait stack-delete-complete --stack-name "$STACK_NAME"
        echo "CloudFormation stack deleted successfully."
    fi

    # Clean up IAM roles
    if aws iam get-role --role-name "$NODE_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from node role: $NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSWorkerNodePolicy --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEC2ContainerRegistryReadOnly --role-name "$NODE_ROLE_NAME"
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKS_CNI_Policy --role-name "$NODE_ROLE_NAME"
        echo "Deleting node role: $NODE_ROLE_NAME"
        aws iam delete-role --role-name "$NODE_ROLE_NAME"
        echo "Node role deleted successfully."
    fi

    if aws iam get-role --role-name "$CLUSTER_ROLE_NAME" 2>/dev/null; then
        echo "Detaching policies from cluster role: $CLUSTER_ROLE_NAME"
```

```
        aws iam detach-role-policy --policy-arn arn:aws:iam::aws:policy/
AmazonEKSClusterPolicy --role-name "$CLUSTER_ROLE_NAME"
        echo "Deleting cluster role: $CLUSTER_ROLE_NAME"
        aws iam delete-role --role-name "$CLUSTER_ROLE_NAME"
        echo "Cluster role deleted successfully."
    fi

    echo "Cleanup complete."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Cleaning up resources..."; cleanup_resources;
 exit 1' SIGINT SIGTERM

# Verify AWS CLI configuration
echo "Verifying AWS CLI configuration..."
AWS_ACCOUNT_INFO=$(aws sts get-caller-identity)
check_command "$AWS_ACCOUNT_INFO"
echo "AWS CLI is properly configured."

# Step 1: Create VPC using CloudFormation
echo "Step 1: Creating VPC with CloudFormation..."
echo "Creating CloudFormation stack: $STACK_NAME"

# Create the CloudFormation stack
CF_CREATE_OUTPUT=$(aws cloudformation create-stack \
  --stack-name "$STACK_NAME" \
  --template-url https://s3.us-west-2.amazonaws.com/amazon-eks/
cloudformation/2020-10-29/amazon-eks-vpc-private-subnets.yaml \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs)
check_command "$CF_CREATE_OUTPUT"
CREATED_RESOURCES+=("CloudFormation Stack: $STACK_NAME")

echo "Waiting for CloudFormation stack to complete (this may take a few
 minutes)..."
aws cloudformation wait stack-create-complete --stack-name "$STACK_NAME"
if [ $? -ne 0 ]; then
    handle_error "CloudFormation stack creation failed"
fi
echo "CloudFormation stack created successfully."

# Step 2: Create IAM roles for EKS
echo "Step 2: Creating IAM roles for EKS..."
```

```
# Create cluster role trust policy
echo "Creating cluster role trust policy..."
cat > eks-cluster-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "eks.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create cluster role
echo "Creating cluster IAM role: $CLUSTER_ROLE_NAME"
CLUSTER_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$CLUSTER_ROLE_NAME" \
  --assume-role-policy-document file://"eks-cluster-role-trust-policy.json")
check_command "$CLUSTER_ROLE_OUTPUT"
aws iam tag-role --role-name "$CLUSTER_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $CLUSTER_ROLE_NAME")

# Attach policy to cluster role
echo "Attaching EKS cluster policy to role..."
ATTACH_CLUSTER_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSClusterPolicy \
  --role-name "$CLUSTER_ROLE_NAME")
check_command "$ATTACH_CLUSTER_POLICY_OUTPUT"

# Create node role trust policy
echo "Creating node role trust policy..."
cat > node-role-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
```

```
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create node role
echo "Creating node IAM role: $NODE_ROLE_NAME"
NODE_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$NODE_ROLE_NAME" \
  --assume-role-policy-document file://"node-role-trust-policy.json")
check_command "$NODE_ROLE_OUTPUT"
aws iam tag-role --role-name "$NODE_ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=eks-gs
CREATED_RESOURCES+=("IAM Role: $NODE_ROLE_NAME")

# Attach policies to node role
echo "Attaching EKS node policies to role..."
ATTACH_NODE_POLICY1_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKSWorkerNodePolicy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY1_OUTPUT"

ATTACH_NODE_POLICY2_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEC2ContainerRegistryReadOnly \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY2_OUTPUT"

ATTACH_NODE_POLICY3_OUTPUT=$(aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:policy/AmazonEKS_CNI_Policy \
  --role-name "$NODE_ROLE_NAME")
check_command "$ATTACH_NODE_POLICY3_OUTPUT"

# Step 3: Get VPC and subnet information
echo "Step 3: Getting VPC and subnet information..."

VPC_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='VpcId'].OutputValue" \
  --output text)
if [ -z "$VPC_ID" ]; then
    handle_error "Failed to get VPC ID from CloudFormation stack"
fi
```

```
echo "VPC ID: $VPC_ID"

SUBNET_IDS=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SubnetIds'].OutputValue" \
  --output text)
if [ -z "$SUBNET_IDS" ]; then
    handle_error "Failed to get Subnet IDs from CloudFormation stack"
fi
echo "Subnet IDs: $SUBNET_IDS"

SECURITY_GROUP_ID=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query "Stacks[0].Outputs[?OutputKey=='SecurityGroups'].OutputValue" \
  --output text)
if [ -z "$SECURITY_GROUP_ID" ]; then
    handle_error "Failed to get Security Group ID from CloudFormation stack"
fi
echo "Security Group ID: $SECURITY_GROUP_ID"

# Step 4: Create EKS cluster
echo "Step 4: Creating EKS cluster: $CLUSTER_NAME"

CLUSTER_ROLE_ARN=$(aws iam get-role --role-name "$CLUSTER_ROLE_NAME" --query
 "Role.Arn" --output text)
if [ -z "$CLUSTER_ROLE_ARN" ]; then
    handle_error "Failed to get Cluster Role ARN"
fi

echo "Creating EKS cluster (this will take 10-15 minutes)..."
CREATE_CLUSTER_OUTPUT=$(aws eks create-cluster \
  --name "$CLUSTER_NAME" \
  --role-arn "$CLUSTER_ROLE_ARN" \
  --resources-vpc-config
 subnetIds="$SUBNET_IDS",securityGroupIds="$SECURITY_GROUP_ID" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_CLUSTER_OUTPUT"
CREATED_RESOURCES+=("EKS Cluster: $CLUSTER_NAME")

echo "Waiting for EKS cluster to become active (this may take 10-15 minutes)..."
aws eks wait cluster-active --name "$CLUSTER_NAME"
if [ $? -ne 0 ]; then
    handle_error "Cluster creation failed or timed out"
fi
```

```
echo "EKS cluster is now active."

# Step 5: Configure kubectl
echo "Step 5: Configuring kubectl to communicate with the cluster..."

# Check if kubectl is installed
if ! check_kubectl; then
    echo "Will skip kubectl configuration steps but continue with the script."
    echo "You can manually configure kubectl later with: aws eks update-
kubeconfig --name \"$CLUSTER_NAME\""
else
    UPDATE_KUBECONFIG_OUTPUT=$(aws eks update-kubeconfig --name "$CLUSTER_NAME")
    check_command "$UPDATE_KUBECONFIG_OUTPUT"
    echo "kubectl configured successfully."

    # Test kubectl configuration
    echo "Testing kubectl configuration..."
    KUBECTL_TEST_OUTPUT=$(kubectl get svc 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: kubectl configuration test failed. This might be due to
 permissions or network issues."
        echo "Error details: $KUBECTL_TEST_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$KUBECTL_TEST_OUTPUT"
        echo "kubectl configuration test successful."
    fi
fi

# Step 6: Create managed node group
echo "Step 6: Creating managed node group: $NODEGROUP_NAME"

NODE_ROLE_ARN=$(aws iam get-role --role-name "$NODE_ROLE_NAME" --query "Role.Arn"
 --output text)
if [ -z "$NODE_ROLE_ARN" ]; then
    handle_error "Failed to get Node Role ARN"
fi

# Convert comma-separated subnet IDs to space-separated for the create-nodegroup
 command
SUBNET_IDS_ARRAY=(${SUBNET_IDS//,/ })

echo "Creating managed node group (this will take 5-10 minutes)..."
CREATE_NODEGROUP_OUTPUT=$(aws eks create-nodegroup \
```

```
  --cluster-name "$CLUSTER_NAME" \
  --nodegroup-name "$NODEGROUP_NAME" \
  --node-role "$NODE_ROLE_ARN" \
  --subnets "${SUBNET_IDS_ARRAY[@]}" \
  --tags Key=project,Value=doc-smith,Key=tutorial,Value=eks-gs)
check_command "$CREATE_NODEGROUP_OUTPUT"
CREATED_RESOURCES+=("EKS Node Group: $NODEGROUP_NAME")

echo "Waiting for node group to become active (this may take 5-10 minutes)..."
aws eks wait nodegroup-active --cluster-name "$CLUSTER_NAME" --nodegroup-name
 "$NODEGROUP_NAME"
if [ $? -ne 0 ]; then
    handle_error "Node group creation failed or timed out"
fi
echo "Node group is now active."

# Step 7: Verify nodes
echo "Step 7: Verifying nodes..."
echo "Waiting for nodes to register with the cluster (this may take a few
 minutes)..."
sleep 60  # Give nodes more time to register

# Check if kubectl is installed before attempting to use it
if ! check_kubectl; then
    echo "Cannot verify nodes without kubectl. Skipping this step."
    echo "You can manually verify nodes after installing kubectl with: kubectl
 get nodes"
else
    NODES_OUTPUT=$(kubectl get nodes 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get nodes. This might be due to permissions or
 the nodes are still registering."
        echo "Error details: $NODES_OUTPUT"
        echo "Continuing with script execution..."
    else
        echo "$NODES_OUTPUT"
        echo "Nodes verified successfully."
    fi
fi

# Step 8: View resources
echo "Step 8: Viewing cluster resources..."

echo "Cluster information:"
```

```
CLUSTER_INFO=$(aws eks describe-cluster --name "$CLUSTER_NAME")
echo "$CLUSTER_INFO"

echo "Node group information:"
NODEGROUP_INFO=$(aws eks describe-nodegroup --cluster-name "$CLUSTER_NAME" --
nodegroup-name "$NODEGROUP_NAME")
echo "$NODEGROUP_INFO"

echo "Kubernetes resources:"
if ! check_kubectl; then
    echo "Cannot list Kubernetes resources without kubectl. Skipping this step."
    echo "You can manually list resources after installing kubectl with: kubectl
 get all --all-namespaces"
else
    KUBE_RESOURCES=$(kubectl get all --all-namespaces 2>&1)
    if [ $? -ne 0 ]; then
        echo "Warning: Unable to get Kubernetes resources. This might be due to
 permissions."
        echo "Error details: $KUBE_RESOURCES"
        echo "Continuing with script execution..."
    else
        echo "$KUBE_RESOURCES"
    fi
fi

# Display summary of created resources
echo ""
echo "========================================="
echo "RESOURCES CREATED"
echo "========================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "========================================="

# Prompt for cleanup
echo ""
echo "========================================="
echo "CLEANUP CONFIRMATION"
echo "========================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
```

```
    cleanup_resources
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run the following commands:"
    echo "1. Delete node group: aws eks delete-nodegroup --cluster-name
 $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "2. Wait for node group deletion: aws eks wait nodegroup-deleted --
cluster-name $CLUSTER_NAME --nodegroup-name $NODEGROUP_NAME"
    echo "3. Delete cluster: aws eks delete-cluster --name $CLUSTER_NAME"
    echo "4. Wait for cluster deletion: aws eks wait cluster-deleted --name
 $CLUSTER_NAME"
    echo "5. Delete CloudFormation stack: aws cloudformation delete-stack --
stack-name $STACK_NAME"
    echo "6. Detach and delete IAM roles for the node group and cluster"
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateCluster
  - CreateNodegroup
  - CreateRole
  - CreateStack
  - DeleteCluster
  - DeleteNodegroup
  - DeleteRole
  - DeleteStack
  - DescribeCluster
  - DescribeNodegroup
  - DescribeStacks
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListNodegroups

- UpdateKubeconfig
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon MSK 入门

以下代码示例展示了如何：

- 创建 MSK 集群
- 为 MSK 访问创建 IAM 权限
- 创建客户端计算机
- 获取引导代理
- 设置客户端计算机
- 创建主题并生成/使用数据
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon MSK Getting Started Tutorial Script - Version 8
# This script automates the steps in the Amazon MSK Getting Started tutorial
# It creates an MSK cluster, sets up IAM permissions, creates a client machine,
# and configures the client to interact with the cluster

# Set up logging
LOG_FILE="msk_tutorial_$(date +%Y%m%d_%H%M%S).log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon MSK Getting Started Tutorial Script - Version 8"
echo "Logging to $LOG_FILE"
echo "============================================="

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Resources created so far:"
    if [ -n "$CLUSTER_ARN" ]; then echo "- MSK Cluster: $CLUSTER_ARN"; fi
    if [ -n "$POLICY_ARN" ]; then echo "- IAM Policy: $POLICY_ARN"; fi
    if [ -n "$ROLE_NAME" ]; then echo "- IAM Role: $ROLE_NAME"; fi
    if [ -n "$INSTANCE_PROFILE_NAME" ]; then echo "- IAM Instance Profile:
 $INSTANCE_PROFILE_NAME"; fi
    if [ -n "$CLIENT_SG_ID" ]; then echo "- Client Security Group:
 $CLIENT_SG_ID"; fi
    if [ -n "$INSTANCE_ID" ]; then echo "- EC2 Instance: $INSTANCE_ID"; fi
    if [ -n "$KEY_NAME" ]; then echo "- Key Pair: $KEY_NAME"; fi

    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check if a resource exists
resource_exists() {
    local resource_type="$1"
    local resource_id="$2"

    case "$resource_type" in
        "cluster")
            aws kafka describe-cluster --cluster-arn "$resource_id" &>/dev/null
            ;;
        "policy")
            aws iam get-policy --policy-arn "$resource_id" &>/dev/null
            ;;
        "role")
            aws iam get-role --role-name "$resource_id" &>/dev/null
            ;;
        "instance-profile")
            aws iam get-instance-profile --instance-profile-name "$resource_id"
 &>/dev/null
            ;;
```

```
        "security-group")
            aws ec2 describe-security-groups --group-ids "$resource_id" &>/dev/
null
            ;;
        "instance")
            aws ec2 describe-instances --instance-ids "$resource_id" --query
 'Reservations[0].Instances[0].State.Name' --output text | grep -v "terminated"
 &>/dev/null
            ;;
        "key-pair")
            aws ec2 describe-key-pairs --key-names "$resource_id" &>/dev/null
            ;;
    esac
}

# Function to remove security group references
remove_security_group_references() {
    local sg_id="$1"

    if [ -z "$sg_id" ]; then
        echo "No security group ID provided for reference removal"
        return
    fi

    echo "Removing security group references for $sg_id"

    # Get all security groups in the VPC that might reference our client security
 group
    local vpc_security_groups=$(aws ec2 describe-security-groups \
        --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" \
        --query 'SecurityGroups[].GroupId' \
        --output text 2>/dev/null)

    if [ -n "$vpc_security_groups" ]; then
        for other_sg in $vpc_security_groups; do
            if [ "$other_sg" != "$sg_id" ]; then
                echo "Checking security group $other_sg for references to $sg_id"

                # Get the security group details in JSON format
                local sg_details=$(aws ec2 describe-security-groups \
                    --group-ids "$other_sg" \
                    --output json 2>/dev/null)

                if [ -n "$sg_details" ]; then
```

```
                    # Check if our security group is referenced in inbound rules
                    local has_inbound_ref=$(echo "$sg_details" | grep -o
 "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_inbound_ref" ]; then
                        echo "Found inbound rules in $other_sg referencing
 $sg_id, removing them..."

                        # Try to remove common rule types
                        echo "Attempting to remove all-traffic rule"
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol all \
                            --source-group "$sg_id" 2>/dev/null || echo "No all-
traffic rule to remove"

                        # Try to remove TCP rules on common ports
                        for port in 22 80 443 9092 9094 9096; do
                            aws ec2 revoke-security-group-ingress \
                                --group-id "$other_sg" \
                                --protocol tcp \
                                --port "$port" \
                                --source-group "$sg_id" 2>/dev/null || true
                        done

                        # Try to remove UDP rules
                        aws ec2 revoke-security-group-ingress \
                            --group-id "$other_sg" \
                            --protocol udp \
                            --source-group "$sg_id" 2>/dev/null || true
                    fi

                    # Check for outbound rules (less common but possible)
                    local has_outbound_ref=$(echo "$sg_details" | grep -A 20
 "IpPermissionsEgress" | grep -o "\"GroupId\": \"$sg_id\"" | head -1)

                    if [ -n "$has_outbound_ref" ]; then
                        echo "Found outbound rules in $other_sg referencing
 $sg_id, removing them..."

                        aws ec2 revoke-security-group-egress \
                            --group-id "$other_sg" \
                            --protocol all \
```

```
                            --source-group "$sg_id" 2>/dev/null || echo "No
 outbound all-traffic rule to remove"
                    fi
                fi
            fi
        done
    fi

    echo "Completed security group reference removal for $sg_id"
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    # Delete EC2 instance if it exists
    if [ -n "$INSTANCE_ID" ] && resource_exists "instance" "$INSTANCE_ID"; then
        echo "Terminating EC2 instance: $INSTANCE_ID"
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || echo "Failed
 to terminate instance"
        echo "Waiting for instance to terminate..."
        aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID" || echo
 "Failed to wait for instance termination"
    fi

    # Delete MSK cluster first (to remove dependencies on security group)
    if [ -n "$CLUSTER_ARN" ] && resource_exists "cluster" "$CLUSTER_ARN"; then
        echo "Deleting MSK cluster: $CLUSTER_ARN"
        aws kafka delete-cluster --cluster-arn "$CLUSTER_ARN" || echo "Failed to
 delete cluster"

        # Wait a bit for the cluster deletion to start
        echo "Waiting 30 seconds for cluster deletion to begin..."
        sleep 30
    fi

    # Remove security group references before attempting deletion
    if [ -n "$CLIENT_SG_ID" ] && resource_exists "security-group"
 "$CLIENT_SG_ID"; then
        remove_security_group_references "$CLIENT_SG_ID"

        echo "Deleting security group: $CLIENT_SG_ID"
        # Try multiple times with longer delays to ensure dependencies are
 removed
```

```
        for i in {1..10}; do
            if aws ec2 delete-security-group --group-id "$CLIENT_SG_ID"; then
                echo "Security group deleted successfully"
                break
            fi
            echo "Failed to delete security group (attempt $i/10), retrying in 30
seconds..."
            sleep 30
        done
    fi

    # Delete key pair if it exists
    if [ -n "$KEY_NAME" ] && resource_exists "key-pair" "$KEY_NAME"; then
        echo "Deleting key pair: $KEY_NAME"
        aws ec2 delete-key-pair --key-name "$KEY_NAME" || echo "Failed to delete
key pair"
    fi

    # Remove role from instance profile
    if [ -n "$ROLE_NAME" ] && [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists
"instance-profile" "$INSTANCE_PROFILE_NAME"; then
        echo "Removing role from instance profile"
        aws iam remove-role-from-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" \
            --role-name "$ROLE_NAME" || echo "Failed to remove role from instance
profile"
    fi

    # Delete instance profile
    if [ -n "$INSTANCE_PROFILE_NAME" ] && resource_exists "instance-profile"
"$INSTANCE_PROFILE_NAME"; then
        echo "Deleting instance profile: $INSTANCE_PROFILE_NAME"
        aws iam delete-instance-profile \
            --instance-profile-name "$INSTANCE_PROFILE_NAME" || echo "Failed to
delete instance profile"
    fi

    # Detach policy from role
    if [ -n "$ROLE_NAME" ] && [ -n "$POLICY_ARN" ] && resource_exists "role"
"$ROLE_NAME"; then
        echo "Detaching policy from role"
        aws iam detach-role-policy \
            --role-name "$ROLE_NAME" \
            --policy-arn "$POLICY_ARN" || echo "Failed to detach policy"
```

```
    fi

    # Delete role
    if [ -n "$ROLE_NAME" ] && resource_exists "role" "$ROLE_NAME"; then
        echo "Deleting role: $ROLE_NAME"
        aws iam delete-role --role-name "$ROLE_NAME" || echo "Failed to delete
 role"
    fi

    # Delete policy
    if [ -n "$POLICY_ARN" ] && resource_exists "policy" "$POLICY_ARN"; then
        echo "Deleting policy: $POLICY_ARN"
        aws iam delete-policy --policy-arn "$POLICY_ARN" || echo "Failed to
 delete policy"
    fi

    echo "Cleanup completed"
}

# Function to find a suitable subnet and instance type combination
find_suitable_subnet_and_instance_type() {
    local vpc_id="$1"
    local -a subnet_array=("${!2}")

    # List of instance types to try, in order of preference
    local instance_types=("t3.micro" "t2.micro" "t3.small" "t2.small")

    echo "Finding suitable subnet and instance type combination..."

    for instance_type in "${instance_types[@]}"; do
        echo "Trying instance type: $instance_type"

        for subnet_id in "${subnet_array[@]}"; do
            # Get the availability zone for this subnet
            local az=$(aws ec2 describe-subnets \
                --subnet-ids "$subnet_id" \
                --query 'Subnets[0].AvailabilityZone' \
                --output text)

            echo "  Checking subnet $subnet_id in AZ $az"

            # Check if this instance type is available in this AZ
            local available=$(aws ec2 describe-instance-type-offerings \
                --location-type availability-zone \
```

```
                --filters "Name=location,Values=$az" "Name=instance-type,Values=
$instance_type" \
                --query 'InstanceTypeOfferings[0].InstanceType' \
                --output text 2>/dev/null)

            if [ "$available" = "$instance_type" ]; then
                echo "  # Found suitable combination: $instance_type in $az
 (subnet: $subnet_id)"
                SELECTED_SUBNET_ID="$subnet_id"
                SELECTED_INSTANCE_TYPE="$instance_type"
                return 0
            else
                echo "  # $instance_type not available in $az"
            fi
        done
    done

    echo "ERROR: Could not find any suitable subnet and instance type
 combination"
    return 1
}

# Generate unique identifiers
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n 1)
CLUSTER_NAME="MSKTutorialCluster-${RANDOM_SUFFIX}"
POLICY_NAME="msk-tutorial-policy-${RANDOM_SUFFIX}"
ROLE_NAME="msk-tutorial-role-${RANDOM_SUFFIX}"
INSTANCE_PROFILE_NAME="msk-tutorial-profile-${RANDOM_SUFFIX}"
SG_NAME="MSKClientSecurityGroup-${RANDOM_SUFFIX}"

echo "Using the following resource names:"
echo "- Cluster Name: $CLUSTER_NAME"
echo "- Policy Name: $POLICY_NAME"
echo "- Role Name: $ROLE_NAME"
echo "- Instance Profile Name: $INSTANCE_PROFILE_NAME"
echo "- Security Group Name: $SG_NAME"
echo "================================================"

# Step 1: Create an MSK Provisioned cluster
echo "Step 1: Creating MSK Provisioned cluster"

# Get the default VPC ID first
echo "Getting default VPC..."
DEFAULT_VPC_ID=$(aws ec2 describe-vpcs \
```

```
    --filters "Name=is-default,Values=true" \
    --query "Vpcs[0].VpcId" \
    --output text)

if [ -z "$DEFAULT_VPC_ID" ] || [ "$DEFAULT_VPC_ID" = "None" ]; then
    handle_error "Could not find default VPC. Please ensure you have a default
 VPC in your region."
fi

echo "Default VPC ID: $DEFAULT_VPC_ID"

# Get available subnets in the default VPC
echo "Getting available subnets in the default VPC..."
SUBNETS=$(aws ec2 describe-subnets \
    --filters "Name=vpc-id,Values=$DEFAULT_VPC_ID" "Name=default-for-
az,Values=true" \
    --query "Subnets[0:3].SubnetId" \
    --output text)

# Convert space-separated subnet IDs to an array
read -r -a SUBNET_ARRAY <<< "$SUBNETS"

if [ ${#SUBNET_ARRAY[@]} -lt 3 ]; then
    handle_error "Not enough subnets available in the default VPC. Need at least
 3 subnets, found ${#SUBNET_ARRAY[@]}."
fi

# Get default security group for the default VPC
echo "Getting default security group for the default VPC..."
DEFAULT_SG=$(aws ec2 describe-security-groups \
    --filters "Name=group-name,Values=default" "Name=vpc-id,Values=
$DEFAULT_VPC_ID" \
    --query "SecurityGroups[0].GroupId" \
    --output text)

if [ -z "$DEFAULT_SG" ] || [ "$DEFAULT_SG" = "None" ]; then
    handle_error "Could not find default security group for VPC $DEFAULT_VPC_ID"
fi

echo "Creating MSK cluster: $CLUSTER_NAME"
echo "Using VPC: $DEFAULT_VPC_ID"
echo "Using subnets: ${SUBNET_ARRAY[0]} ${SUBNET_ARRAY[1]} ${SUBNET_ARRAY[2]}"
echo "Using security group: $DEFAULT_SG"
```

```
# Create the MSK cluster with proper error handling
CLUSTER_RESPONSE=$(aws kafka create-cluster \
    --cluster-name "$CLUSTER_NAME" \
    --broker-node-group-info "{\"InstanceType\": \"kafka.t3.small\",
 \"ClientSubnets\": [\"${SUBNET_ARRAY[0]}\", \"${SUBNET_ARRAY[1]}\",
 \"${SUBNET_ARRAY[2]}\"], \"SecurityGroups\": [\"$DEFAULT_SG\"]}" \
    --kafka-version "3.6.0" \
    --number-of-broker-nodes 3 \
    --encryption-info "{\"EncryptionInTransit\": {\"InCluster\": true,
 \"ClientBroker\": \"TLS\"}}" \
    --tags project=doc-smith,tutorial=amazon-managed-streaming-for-apache-kafka-
gs 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create MSK cluster: $CLUSTER_RESPONSE"
fi

# Extract the cluster ARN using grep
CLUSTER_ARN=$(echo "$CLUSTER_RESPONSE" | grep -o '"ClusterArn": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLUSTER_ARN" ]; then
    handle_error "Failed to extract cluster ARN from response: $CLUSTER_RESPONSE"
fi

echo "MSK cluster creation initiated. ARN: $CLUSTER_ARN"
echo "Waiting for cluster to become active (this may take 15-20 minutes)..."

# Wait for the cluster to become active
while true; do
    CLUSTER_STATUS=$(aws kafka describe-cluster --cluster-arn "$CLUSTER_ARN" --
query "ClusterInfo.State" --output text 2>/dev/null)

    if [ $? -ne 0 ]; then
        echo "Failed to get cluster status. Retrying in 30 seconds..."
        sleep 30
        continue
    fi

    echo "Current cluster status: $CLUSTER_STATUS"

    if [ "$CLUSTER_STATUS" = "ACTIVE" ]; then
        echo "Cluster is now active!"
```

```
        break
    elif [ "$CLUSTER_STATUS" = "FAILED" ]; then
        handle_error "Cluster creation failed"
    fi

    echo "Still waiting for cluster to become active... (checking again in 60
 seconds)"
    sleep 60
done

echo "============================================="
# Step 2: Create an IAM role granting access to create topics on the Amazon MSK
 cluster
echo "Step 2: Creating IAM policy and role"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION=$(aws ec2 describe-availability-zones --query
 'AvailabilityZones[0].RegionName' --output text)
fi

if [ -z "$ACCOUNT_ID" ] || [ -z "$REGION" ]; then
    handle_error "Could not determine AWS account ID or region"
fi

echo "Account ID: $ACCOUNT_ID"
echo "Region: $REGION"

# Create IAM policy
echo "Creating IAM policy: $POLICY_NAME"
POLICY_DOCUMENT="{
    \"Version\": \"2012-10-17\",
    \"Statement\": [
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:Connect\",
                \"kafka-cluster:AlterCluster\",
                \"kafka-cluster:DescribeCluster\"
            ],
            \"Resource\": [
                \"$CLUSTER_ARN\"
```

```
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:*Topic*\",
                \"kafka-cluster:WriteData\",
                \"kafka-cluster:ReadData\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:topic/$CLUSTER_NAME/*\"
            ]
        },
        {
            \"Effect\": \"Allow\",
            \"Action\": [
                \"kafka-cluster:AlterGroup\",
                \"kafka-cluster:DescribeGroup\"
            ],
            \"Resource\": [
                \"arn:aws:kafka:$REGION:$ACCOUNT_ID:group/$CLUSTER_NAME/*\"
            ]
        }
    ]
}"

POLICY_RESPONSE=$(aws iam create-policy \
    --policy-name "$POLICY_NAME" \
    --policy-document "$POLICY_DOCUMENT" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM policy: $POLICY_RESPONSE"
fi

# Extract the policy ARN using grep
POLICY_ARN=$(echo "$POLICY_RESPONSE" | grep -o '"Arn": "[^"]*' | cut -d'"' -f4)

if [ -z "$POLICY_ARN" ]; then
    handle_error "Failed to extract policy ARN from response: $POLICY_RESPONSE"
fi

aws iam tag-policy --policy-arn "$POLICY_ARN" \
```

```
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

echo "IAM policy created. ARN: $POLICY_ARN"

# Create IAM role for EC2
echo "Creating IAM role: $ROLE_NAME"
TRUST_POLICY="{\"Version\":\"2012-10-17\",\"Statement\":[{\"Effect\":\"Allow\",
\"Principal\":{\"Service\":\"ec2.amazonaws.com\"},\"Action\":\"sts:AssumeRole
\"}]}"

ROLE_RESPONSE=$(aws iam create-role \
    --role-name "$ROLE_NAME" \
    --assume-role-policy-document "$TRUST_POLICY" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create IAM role: $ROLE_RESPONSE"
fi

echo "IAM role created: $ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs

# Attach policy to role
echo "Attaching policy to role"
ATTACH_RESPONSE=$(aws iam attach-role-policy \
    --role-name "$ROLE_NAME" \
    --policy-arn "$POLICY_ARN" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to attach policy to role: $ATTACH_RESPONSE"
fi

echo "Policy attached to role"

# Create instance profile and add role to it
echo "Creating instance profile: $INSTANCE_PROFILE_NAME"
PROFILE_RESPONSE=$(aws iam create-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" 2>&1)

# Check if the command was successful
```

```
if [ $? -ne 0 ]; then
    handle_error "Failed to create instance profile: $PROFILE_RESPONSE"
fi

echo "Instance profile created"

echo "Adding role to instance profile"
ADD_ROLE_RESPONSE=$(aws iam add-role-to-instance-profile \
    --instance-profile-name "$INSTANCE_PROFILE_NAME" \
    --role-name "$ROLE_NAME" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to add role to instance profile: $ADD_ROLE_RESPONSE"
fi

echo "Role added to instance profile"

# Wait a moment for IAM propagation
echo "Waiting 10 seconds for IAM propagation..."
sleep 10

echo "============================================="

# Step 3: Create a client machine
echo "Step 3: Creating client machine"

# Find a suitable subnet and instance type combination
if ! find_suitable_subnet_and_instance_type "$DEFAULT_VPC_ID" SUBNET_ARRAY[@];
 then
    handle_error "Could not find a suitable subnet and instance type combination"
fi

echo "Selected subnet: $SELECTED_SUBNET_ID"
echo "Selected instance type: $SELECTED_INSTANCE_TYPE"

# Verify the subnet is in the same VPC we're using
SUBNET_VPC_ID=$(aws ec2 describe-subnets \
    --subnet-ids "$SELECTED_SUBNET_ID" \
    --query 'Subnets[0].VpcId' \
    --output text)

if [ "$SUBNET_VPC_ID" != "$DEFAULT_VPC_ID" ]; then
```

```
    handle_error "Subnet VPC ($SUBNET_VPC_ID) does not match default VPC
 ($DEFAULT_VPC_ID)"
fi

echo "VPC ID: $SUBNET_VPC_ID"

# Get security group ID from the MSK cluster
echo "Getting security group ID from the MSK cluster"
MSK_SG_ID=$(aws kafka describe-cluster \
    --cluster-arn "$CLUSTER_ARN" \
    --query 'ClusterInfo.BrokerNodeGroupInfo.SecurityGroups[0]' \
    --output text)

if [ -z "$MSK_SG_ID" ] || [ "$MSK_SG_ID" = "None" ]; then
    handle_error "Failed to get security group ID from cluster"
fi

echo "MSK security group ID: $MSK_SG_ID"

# Create security group for client
echo "Creating security group for client: $SG_NAME"
CLIENT_SG_RESPONSE=$(aws ec2 create-security-group \
    --group-name "$SG_NAME" \
    --description "Security group for MSK client" \
    --vpc-id "$DEFAULT_VPC_ID" \
    --tag-specifications 'ResourceType=security-
group,Tags=[{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-managed-
streaming-for-apache-kafka-gs}]' 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create security group: $CLIENT_SG_RESPONSE"
fi

# Extract the security group ID using grep
CLIENT_SG_ID=$(echo "$CLIENT_SG_RESPONSE" | grep -o '"GroupId": "[^"]*' | cut -
d'"' -f4)

if [ -z "$CLIENT_SG_ID" ]; then
    handle_error "Failed to extract security group ID from response:
 $CLIENT_SG_RESPONSE"
fi

echo "Client security group created. ID: $CLIENT_SG_ID"
```

```
# Allow SSH access to client from your IP only
echo "Getting your public IP address"
MY_IP=$(curl -s https://checkip.amazonaws.com 2>/dev/null)

if [ -z "$MY_IP" ]; then
    echo "Warning: Could not determine your IP address. Using 0.0.0.0/0 (not
 recommended for production)"
    MY_IP="0.0.0.0/0"
else
    MY_IP="$MY_IP/32"
    echo "Your IP address: $MY_IP"
fi

echo "Adding SSH ingress rule to client security group"
SSH_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$CLIENT_SG_ID" \
    --protocol tcp \
    --port 22 \
    --cidr "$MY_IP" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add SSH ingress rule: $SSH_RULE_RESPONSE"
    echo "You may need to manually add SSH access to security group
 $CLIENT_SG_ID"
fi

echo "SSH ingress rule added"

# Update MSK security group to allow traffic from client security group
echo "Adding ingress rule to MSK security group to allow traffic from client"
MSK_RULE_RESPONSE=$(aws ec2 authorize-security-group-ingress \
    --group-id "$MSK_SG_ID" \
    --protocol all \
    --source-group "$CLIENT_SG_ID" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    echo "Warning: Failed to add ingress rule to MSK security group:
 $MSK_RULE_RESPONSE"
    echo "You may need to manually add ingress rule to security group $MSK_SG_ID"
fi
```

```
echo "Ingress rule added to MSK security group"

# Create key pair
KEY_NAME="MSKKeyPair-${RANDOM_SUFFIX}"
echo "Creating key pair: $KEY_NAME"
KEY_RESPONSE=$(aws ec2 create-key-pair --key-name "$KEY_NAME" --tag-
specifications 'ResourceType=key-pair,Tags=[{Key=project,Value=doc-smith},
{Key=tutorial,Value=amazon-managed-streaming-for-apache-kafka-gs}]' --query
 'KeyMaterial' --output text 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to create key pair: $KEY_RESPONSE"
fi

# Save the private key to a file
KEY_FILE="${KEY_NAME}.pem"
echo "$KEY_RESPONSE" > "$KEY_FILE"
chmod 400 "$KEY_FILE"

echo "Key pair created and saved to $KEY_FILE"

# Get the latest Amazon Linux 2 AMI
echo "Getting latest Amazon Linux 2 AMI ID"
AMI_ID=$(aws ec2 describe-images \
    --owners amazon \
    --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
    --query "sort_by(Images, &CreationDate)[-1].ImageId" \
    --output text 2>/dev/null)

if [ -z "$AMI_ID" ] || [ "$AMI_ID" = "None" ]; then
    handle_error "Failed to get Amazon Linux 2 AMI ID"
fi

echo "Using AMI ID: $AMI_ID"

# Launch EC2 instance with the selected subnet and instance type
echo "Launching EC2 instance"
echo "Instance type: $SELECTED_INSTANCE_TYPE"
echo "Subnet: $SELECTED_SUBNET_ID"

INSTANCE_RESPONSE=$(aws ec2 run-instances \
    --image-id "$AMI_ID" \
```

```
    --instance-type "$SELECTED_INSTANCE_TYPE" \
    --key-name "$KEY_NAME" \
    --security-group-ids "$CLIENT_SG_ID" \
    --subnet-id "$SELECTED_SUBNET_ID" \
    --iam-instance-profile "Name=$INSTANCE_PROFILE_NAME" \
    --tag-specifications
 "ResourceType=instance,Tags=[{Key=Name,Value=MSKTutorialClient-
${RANDOM_SUFFIX}},{Key=project,Value=doc-smith},{Key=tutorial,Value=amazon-
managed-streaming-for-apache-kafka-gs}]" 2>&1)

# Check if the command was successful
if [ $? -ne 0 ]; then
    handle_error "Failed to launch EC2 instance: $INSTANCE_RESPONSE"
fi

# Extract the instance ID using grep
INSTANCE_ID=$(echo "$INSTANCE_RESPONSE" | grep -o '"InstanceId": "[^"]*' | head
 -1 | cut -d'"' -f4)

if [ -z "$INSTANCE_ID" ]; then
    handle_error "Failed to extract instance ID from response:
 $INSTANCE_RESPONSE"
fi

echo "EC2 instance launched successfully. ID: $INSTANCE_ID"
echo "Waiting for instance to be running..."

# Wait for the instance to be running
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"

if [ $? -ne 0 ]; then
    handle_error "Instance failed to reach running state"
fi

# Wait a bit more for the instance to initialize
echo "Instance is running. Waiting 30 seconds for initialization..."
sleep 30

# Get public DNS name of instance
CLIENT_DNS=$(aws ec2 describe-instances \
    --instance-ids "$INSTANCE_ID" \
    --query 'Reservations[0].Instances[0].PublicDnsName' \
    --output text)
```

```
if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
    echo "Warning: Could not get public DNS name for instance. Trying public
 IP..."
    CLIENT_DNS=$(aws ec2 describe-instances \
        --instance-ids "$INSTANCE_ID" \
        --query 'Reservations[0].Instances[0].PublicIpAddress' \
        --output text)

    if [ -z "$CLIENT_DNS" ] || [ "$CLIENT_DNS" = "None" ]; then
        handle_error "Failed to get public DNS name or IP address for instance"
    fi
fi

echo "Client instance DNS/IP: $CLIENT_DNS"
echo "============================================="
# Get bootstrap brokers with improved logic
echo "Getting bootstrap brokers"
MAX_RETRIES=10
RETRY_COUNT=0
BOOTSTRAP_BROKERS=""
AUTH_METHOD=""

while [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; do
    # Get the full bootstrap brokers response
    BOOTSTRAP_RESPONSE=$(aws kafka get-bootstrap-brokers \
        --cluster-arn "$CLUSTER_ARN" 2>/dev/null)

    if [ $? -eq 0 ] && [ -n "$BOOTSTRAP_RESPONSE" ]; then
        # Try to get IAM authentication brokers first using grep
        BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringSaslIam": "[^"]*' | cut -d'"' -f4)
        if [ -n "$BOOTSTRAP_BROKERS" ]; then
            AUTH_METHOD="IAM"
        else
            # Fall back to TLS authentication
            BOOTSTRAP_BROKERS=$(echo "$BOOTSTRAP_RESPONSE" | grep -o
 '"BootstrapBrokerStringTls": "[^"]*' | cut -d'"' -f4)
            if [ -n "$BOOTSTRAP_BROKERS" ]; then
                AUTH_METHOD="TLS"
            fi
        fi
    fi

    RETRY_COUNT=$((RETRY_COUNT + 1))
```

```
    if [ "$RETRY_COUNT" -ge "$MAX_RETRIES" ]; then
        echo "Warning: Could not get bootstrap brokers after $MAX_RETRIES
 attempts."
        echo "You may need to manually retrieve them later using:"
        echo "aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
        BOOTSTRAP_BROKERS="BOOTSTRAP_BROKERS_NOT_AVAILABLE"
        AUTH_METHOD="UNKNOWN"
        break
    fi

    if [ -z "$BOOTSTRAP_BROKERS" ] || [ "$BOOTSTRAP_BROKERS" = "None" ]; then
        echo "Bootstrap brokers not available yet. Retrying in 30 seconds...
 (Attempt $RETRY_COUNT/$MAX_RETRIES)"
        sleep 30
    fi
done

echo "Bootstrap brokers: $BOOTSTRAP_BROKERS"
echo "Authentication method: $AUTH_METHOD"
echo "============================================="

# Create setup script for the client machine
echo "Creating setup script for the client machine"
cat > setup_client.sh << 'EOF'
#!/bin/bash

# Set up logging
LOG_FILE="client_setup_$(date +%Y%m%d_%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting client setup"
echo "============================================="

# Install Java
echo "Installing Java"
sudo yum -y install java-11

# Set environment variables
echo "Setting up environment variables"
export KAFKA_VERSION="3.6.0"
echo "KAFKA_VERSION=$KAFKA_VERSION"

# Download and extract Apache Kafka
```

```
echo "Downloading Apache Kafka"
wget https://archive.apache.org/dist/kafka/$KAFKA_VERSION/kafka_2.13-
$KAFKA_VERSION.tgz
if [ $? -ne 0 ]; then
    echo "Failed to download Kafka. Trying alternative mirror..."
    wget https://www.apache.org/dyn/closer.cgi?path=/kafka/$KAFKA_VERSION/
kafka_2.13-$KAFKA_VERSION.tgz
fi

echo "Extracting Kafka"
tar -xzf kafka_2.13-$KAFKA_VERSION.tgz
export KAFKA_ROOT=$(pwd)/kafka_2.13-$KAFKA_VERSION
echo "KAFKA_ROOT=$KAFKA_ROOT"

# Download the MSK IAM authentication package (needed for both IAM and TLS)
echo "Downloading MSK IAM authentication package"
cd $KAFKA_ROOT/libs
wget https://github.com/aws/aws-msk-iam-auth/releases/latest/download/aws-msk-
iam-auth-1.1.6-all.jar
if [ $? -ne 0 ]; then
    echo "Failed to download specific version. Trying to get latest version..."
    LATEST_VERSION=$(curl -s https://api.github.com/repos/aws/aws-msk-iam-auth/
releases/latest | grep -o '"tag_name": "[^"]*' | cut -d'"' -f4)
    wget https://github.com/aws/aws-msk-iam-auth/releases/download/
$LATEST_VERSION/aws-msk-iam-auth-$LATEST_VERSION-all.jar
    if [ $? -ne 0 ]; then
        echo "Failed to download IAM auth package. Please check the URL and try
 again."
        exit 1
    fi
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-$LATEST_VERSION-all.jar
else
    export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
fi
echo "CLASSPATH=$CLASSPATH"

# Create client properties file based on authentication method
echo "Creating client properties file"
cd $KAFKA_ROOT/config

# The AUTH_METHOD_PLACEHOLDER will be replaced by the script
AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

if [ "$AUTH_METHOD" = "IAM" ]; then
```

```
    echo "Configuring for IAM authentication"
    cat > client.properties << 'EOT'
security.protocol=SASL_SSL
sasl.mechanism=AWS_MSK_IAM
sasl.jaas.config=software.amazon.msk.auth.iam.IAMLoginModule required;
sasl.client.callback.handler.class=software.amazon.msk.auth.iam.IAMClientCallbackHandler
EOT
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "Configuring for TLS authentication"
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
else
    echo "Unknown authentication method. Creating basic TLS configuration."
    cat > client.properties << 'EOT'
security.protocol=SSL
EOT
fi

echo "Client setup completed"
echo "============================================="

# Create a script to set up environment variables
cat > ~/setup_env.sh << 'EOT'
#!/bin/bash
export KAFKA_VERSION="3.6.0"
export KAFKA_ROOT=~/kafka_2.13-$KAFKA_VERSION
export CLASSPATH=$KAFKA_ROOT/libs/aws-msk-iam-auth-1.1.6-all.jar
export BOOTSTRAP_SERVER="BOOTSTRAP_SERVER_PLACEHOLDER"
export AUTH_METHOD="AUTH_METHOD_PLACEHOLDER"

echo "Environment variables set:"
echo "KAFKA_VERSION=$KAFKA_VERSION"
echo "KAFKA_ROOT=$KAFKA_ROOT"
echo "CLASSPATH=$CLASSPATH"
echo "BOOTSTRAP_SERVER=$BOOTSTRAP_SERVER"
echo "AUTH_METHOD=$AUTH_METHOD"
EOT

chmod +x ~/setup_env.sh

echo "Created environment setup script: ~/setup_env.sh"
echo "Run 'source ~/setup_env.sh' to set up your environment"
EOF
```

```
# Replace placeholders in the setup script
if [ -n "$BOOTSTRAP_BROKERS" ] && [ "$BOOTSTRAP_BROKERS" != "None" ] &&
 [ "$BOOTSTRAP_BROKERS" != "BOOTSTRAP_BROKERS_NOT_AVAILABLE" ]; then
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|$BOOTSTRAP_BROKERS|g" setup_client.sh
else
    # If bootstrap brokers are not available, provide instructions to get them
    sed -i "s|BOOTSTRAP_SERVER_PLACEHOLDER|\$(aws kafka get-bootstrap-brokers
 --cluster-arn $CLUSTER_ARN --query 'BootstrapBrokerStringTls' --output text)|g"
 setup_client.sh
fi

# Replace auth method placeholder
sed -i "s|AUTH_METHOD_PLACEHOLDER|$AUTH_METHOD|g" setup_client.sh

echo "Setup script created"
echo "============================================="

# Display summary of created resources
echo ""
echo "============================================="
echo "RESOURCE SUMMARY"
echo "============================================="
echo "MSK Cluster ARN: $CLUSTER_ARN"
echo "MSK Cluster Name: $CLUSTER_NAME"
echo "Authentication Method: $AUTH_METHOD"
echo "IAM Policy ARN: $POLICY_ARN"
echo "IAM Role Name: $ROLE_NAME"
echo "IAM Instance Profile: $INSTANCE_PROFILE_NAME"
echo "Client Security Group: $CLIENT_SG_ID"
echo "EC2 Instance ID: $INSTANCE_ID"
echo "EC2 Instance Type: $SELECTED_INSTANCE_TYPE"
echo "EC2 Instance DNS: $CLIENT_DNS"
echo "Key Pair: $KEY_NAME (saved to $KEY_FILE)"
echo "Bootstrap Brokers: $BOOTSTRAP_BROKERS"
echo "============================================="
echo ""

# Instructions for connecting to the instance and setting up the client
echo "NEXT STEPS:"
echo "1. Connect to your EC2 instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS"
echo ""
echo "2. Upload the setup script to your instance:"
```

```
echo "   scp -i $KEY_FILE setup_client.sh ec2-user@$CLIENT_DNS:~/"
echo ""
echo "3. Run the setup script on your instance:"
echo "   ssh -i $KEY_FILE ec2-user@$CLIENT_DNS 'chmod +x ~/setup_client.sh && ~/
setup_client.sh'"
echo ""
echo "4. Source the environment setup script:"
echo "   source ~/setup_env.sh"
echo ""

# Provide different instructions based on authentication method
if [ "$AUTH_METHOD" = "IAM" ]; then
    echo "5. Create a Kafka topic (using IAM authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
elif [ "$AUTH_METHOD" = "TLS" ]; then
    echo "5. Create a Kafka topic (using TLS authentication):"
    echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --create \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --command-config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --replication-factor 3 \\"
    echo "     --partitions 1 \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "6. Start a producer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-producer.sh \\"
    echo "     --broker-list \$BOOTSTRAP_SERVER \\"
```

```
    echo "     --producer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic"
    echo ""
    echo "7. Start a consumer:"
    echo "   \$KAFKA_ROOT/bin/kafka-console-consumer.sh \\"
    echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
    echo "     --consumer.config \$KAFKA_ROOT/config/client.properties \\"
    echo "     --topic MSKTutorialTopic \\"
    echo "     --from-beginning"
else
    echo "5. Manually retrieve bootstrap brokers and configure authentication:"
    echo "   aws kafka get-bootstrap-brokers --cluster-arn $CLUSTER_ARN"
fi

echo ""
echo "8. Verify the topic was created:"
echo "   \$KAFKA_ROOT/bin/kafka-topics.sh --list \\"
echo "     --bootstrap-server \$BOOTSTRAP_SERVER \\"
echo "     --command-config \$KAFKA_ROOT/config/client.properties"
echo "================================================"
echo ""

# Ask if user wants to clean up resources
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    cleanup_resources
    echo "All resources have been cleaned up."
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
    echo "To clean up resources, run this script again and choose 'y' when
 prompted."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。

- AddRoleToInstanceProfile
- AttachRolePolicy
- AuthorizeSecurityGroupIngress
- CreateCluster
- CreateInstanceProfile
- CreateKeyPair
- CreatePolicy
- CreateRole
- CreateSecurityGroup
- DeleteCluster
- DeleteInstanceProfile
- DeleteKeyPair
- DeletePolicy
- DeleteRole
- DeleteSecurityGroup
- DescribeAvailabilityZones
- DescribeCluster
- DescribeImages
- DescribeInstanceTypeOfferings
- DescribeInstances
- DescribeKeyPairs
- DescribeSecurityGroups
- DescribeSubnets
- DescribeVpcs
- DetachRolePolicy
- GetBootstrapBrokers
- GetCallerIdentity
- GetInstanceProfile
- GetPolicy
- GetRole

- RemoveRoleFromInstanceProfile
- RevokeSecurityGroupEgress
- RevokeSecurityGroupIngress
- RunInstances
- TerminateInstances
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon OpenSearch Service 入门

以下代码示例展示了如何：

- 创建 OpenSearch Service 域
- 将数据上载到您的域
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon OpenSearch Service Getting Started Script - Version 8 Fixed
# This script creates an OpenSearch domain, uploads data, searches documents, and
 cleans up resources
# Based on the tested and working 4-tutorial-final.md

# SECURITY IMPROVEMENTS IN THIS VERSION:
# 1. Removed hardcoded passwords - now generated securely
```

```
# 2. Improved access policy to use principle of least privilege
# 3. Added credential masking in logs
# 4. Removed credentials from command output
# 5. Added input validation for AWS region
# 6. Improved error handling and validation
# 7. Added secure temporary file handling
# 8. Removed unnecessary curl insecurity flags
# 9. Added better secret management practices
# 10. Improved resource tagging for better governance

set -e  # Exit on any error

# Set up logging with restricted permissions
LOG_FILE="opensearch_tutorial_v8_fixed.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting Amazon OpenSearch Service tutorial script v8-fixed at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"

# Track if domain was successfully created
DOMAIN_CREATED=false
DOMAIN_ACTIVE=false

# Secure temporary directory
TEMP_DIR=$(mktemp -d)
trap 'rm -rf "$TEMP_DIR"' EXIT
chmod 700 "$TEMP_DIR"

# Error handling function
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to clean up resources
cleanup_resources() {
    echo "Cleaning up resources..."

    if [[ "$DOMAIN_CREATED" == "true" ]]; then
```

```
        echo "Checking if domain $DOMAIN_NAME exists before attempting to
 delete..."

        # Check if domain exists before trying to delete
        if aws opensearch describe-domain --domain-name "$DOMAIN_NAME" > /dev/
null 2>&1; then
            echo "Domain $DOMAIN_NAME exists. Proceeding with deletion."
            aws opensearch delete-domain --domain-name "$DOMAIN_NAME"
            echo "Domain deletion initiated. This may take several minutes to
 complete."
        else
            echo "Domain $DOMAIN_NAME does not exist or is not accessible. No
 deletion needed."
        fi
    else
        echo "No domain was successfully created. Nothing to clean up."
    fi

    # Securely clean up sensitive files
    if [[ -d "$TEMP_DIR" ]]; then
        shred -vfz -n 3 "$TEMP_DIR"/* 2>/dev/null || true
        rm -rf "$TEMP_DIR"
    fi
}

# Generate a random identifier for resource names to avoid conflicts
RANDOM_ID=$(openssl rand -hex 4)
DOMAIN_NAME="movies-${RANDOM_ID}"
MASTER_USER="master-user"

# Generate secure password using openssl instead of hardcoding
MASTER_PASSWORD=$(openssl rand -base64 32 | tr -d "=+/" | cut -c1-25)

# Mask password in logs
echo "Using domain name: $DOMAIN_NAME"
echo "Using master username: $MASTER_USER"
echo "Using master password: [REDACTED]"

# Get AWS account ID (matches tutorial)
echo "Retrieving AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
if [[ $? -ne 0 ]] || [[ -z "$ACCOUNT_ID" ]]; then
    handle_error "Failed to retrieve AWS account ID. Please check your AWS
 credentials."
```

```
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Get current region (matches tutorial)
echo "Retrieving current AWS region..."
AWS_REGION=$(aws configure get region)
if [[ -z "$AWS_REGION" ]]; then
    AWS_REGION="us-east-1"
    echo "No region found in AWS config, defaulting to $AWS_REGION"
else
    echo "Using AWS region: $AWS_REGION"
fi

# Validate AWS region format
if ! [[ "$AWS_REGION" =~ ^[a-z]{2}-[a-z]+-[0-9]$ ]]; then
    handle_error "Invalid AWS region format: $AWS_REGION"
fi

# Step 1: Create an OpenSearch Service Domain
echo "Creating OpenSearch Service domain..."
echo "This may take 15-30 minutes to complete."

# SECURITY IMPROVED: Use least privilege access policy
# This policy restricts access to specific actions and should be further
 restricted to specific principals in production
# Create the domain using CloudFormation (handles the 15-20 min creation wait)
SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
STACK_NAME="tutorial-opensearch-${RANDOM_ID}"

echo "Creating domain $DOMAIN_NAME via CloudFormation..."
echo "This typically takes 15-20 minutes. CloudFormation will wait for
 completion."
aws cloudformation deploy \
  --template-file "$SCRIPT_DIR/cfn-opensearch-domain.yaml" \
  --stack-name "$STACK_NAME" \
  --parameter-overrides \
    DomainName="$DOMAIN_NAME" \
    MasterUserName="$MASTER_USER" \
    MasterUserPassword="$MASTER_PASSWORD" \
  --tags project=doc-smith tutorial=opensearch-service-gs \
  --no-fail-on-empty-changeset 2>&1 || handle_error "CloudFormation stack
 creation failed"

DOMAIN_CREATED=true
```

```
echo "Domain created successfully via CloudFormation."

# Get domain endpoint from stack outputs
DOMAIN_ENDPOINT=$(aws cloudformation describe-stacks \
  --stack-name "$STACK_NAME" \
  --query 'Stacks[0].Outputs[?OutputKey==`DomainEndpoint`].OutputValue' \
  --output text)

if [[ -z "$DOMAIN_ENDPOINT" ]] || [[ "$DOMAIN_ENDPOINT" == "None" ]]; then
    handle_error "Failed to get domain endpoint from CloudFormation outputs"
fi

echo "Domain endpoint: $DOMAIN_ENDPOINT"

# Wait for fine-grained access control to be fully ready
echo "Waiting for fine-grained access control to initialize..."
sleep 120

# Verify variables are set correctly (matches tutorial)
echo "Verifying configuration..."
echo "Domain endpoint: $DOMAIN_ENDPOINT"
echo "Master user: $MASTER_USER"
echo "Password set: $(if [ -n "$MASTER_PASSWORD" ]; then echo "Yes"; else echo
 "No"; fi)"

# Step 2: Upload Data to the Domain
echo "Preparing to upload data to the domain..."

# Create a file for the single document (matches tutorial exactly)
echo "Creating single document JSON file..."
SINGLE_MOVIE_FILE="$TEMP_DIR/single_movie.json"
cat > "$SINGLE_MOVIE_FILE" << 'EOF'
{
  "director": "Burton, Tim",
  "genre": ["Comedy","Sci-Fi"],
  "year": 1996,
  "actor": ["Jack Nicholson","Pierce Brosnan","Sarah Jessica Parker"],
  "title": "Mars Attacks!"
}
EOF
chmod 600 "$SINGLE_MOVIE_FILE"

# Create a file for bulk documents (matches tutorial exactly)
echo "Creating bulk documents JSON file..."
```

```
BULK_MOVIES_FILE="$TEMP_DIR/bulk_movies.json"
cat > "$BULK_MOVIES_FILE" << 'EOF'
{ "index" : { "_index": "movies", "_id" : "2" } }
{"director": "Frankenheimer, John", "genre": ["Drama", "Mystery", "Thriller",
 "Crime"], "year": 1962, "actor": ["Lansbury, Angela", "Sinatra, Frank", "Leigh,
 Janet", "Harvey, Laurence", "Silva, Henry", "Frees, Paul", "Gregory, James",
 "Bissell, Whit", "McGiver, John", "Parrish, Leslie", "Edwards, James", "Flowers,
 Bess", "Dhiegh, Khigh", "Payne, Julie", "Kleeb, Helen", "Gray, Joe", "Nalder,
 Reggie", "Stevens, Bert", "Masters, Michael", "Lowell, Tom"], "title": "The
 Manchurian Candidate"}
{ "index" : { "_index": "movies", "_id" : "3" } }
{"director": "Baird, Stuart", "genre": ["Action", "Crime", "Thriller"], "year":
 1998, "actor": ["Downey Jr., Robert", "Jones, Tommy Lee", "Snipes, Wesley",
 "Pantoliano, Joe", "Jacob, Irène", "Nelligan, Kate", "Roebuck, Daniel",
 "Malahide, Patrick", "Richardson, LaTanya", "Wood, Tom", "Kosik, Thomas",
 "Stellate, Nick", "Minkoff, Robert", "Brown, Spitfire", "Foster, Reese",
 "Spielbauer, Bruce", "Mukherji, Kevin", "Cray, Ed", "Fordham, David", "Jett,
 Charlie"], "title": "U.S. Marshals"}
{ "index" : { "_index": "movies", "_id" : "4" } }
{"director": "Ray, Nicholas", "genre": ["Drama", "Romance"], "year": 1955,
 "actor": ["Hopper, Dennis", "Wood, Natalie", "Dean, James", "Mineo, Sal",
 "Backus, Jim", "Platt, Edward", "Ray, Nicholas", "Hopper, William", "Allen,
 Corey", "Birch, Paul", "Hudson, Rochelle", "Doran, Ann", "Hicks, Chuck", "Leigh,
 Nelson", "Williams, Robert", "Wessel, Dick", "Bryar, Paul", "Sessions, Almira",
 "McMahon, David", "Peters Jr., House"], "title": "Rebel Without a Cause"}
EOF
chmod 600 "$BULK_MOVIES_FILE"

# Check if curl is installed
if ! command -v curl &> /dev/null; then
    echo "Warning: curl is not installed. Skipping data upload and search steps."
    echo "You can manually upload the data later using the commands in the
 tutorial."
else
    # IMPROVED: Test authentication with multiple approaches
    echo "Testing authentication with the OpenSearch domain..."
    echo "This test checks if fine-grained access control is ready for data
 operations."

    # Create credentials file for secure handling
    CREDENTIALS_FILE="$TEMP_DIR/.credentials"
    echo "$MASTER_USER:$MASTER_PASSWORD" > "$CREDENTIALS_FILE"
    chmod 600 "$CREDENTIALS_FILE"
```

```
    # Test 1: Basic authentication with root endpoint
    echo "Testing basic authentication with root endpoint..."
    AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
        --netrc-file "$CREDENTIALS_FILE" \
        --request GET \
        "https://${DOMAIN_ENDPOINT}/" 2>&1)

    echo "Basic auth test result:"
    echo "$AUTH_TEST_RESULT"

    # Extract HTTP status code
    HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)

    # Function to check if HTTP code is 2xx
    is_success_code() {
        local code=$1
        if [[ "$code" =~ ^2[0-9][0-9]$ ]]; then
            return 0
        else
            return 1
        fi
    }

    # Check if basic authentication test was successful (200 or 2xx status codes)
    if is_success_code "$HTTP_CODE"; then
        echo "# Basic authentication test successful! (HTTP $HTTP_CODE)"
        AUTH_SUCCESS=true
        AUTH_METHOD="basic"
    else
        echo "Basic authentication failed with HTTP code: $HTTP_CODE"

        # Test 2: Try cluster health endpoint
        echo "Testing with cluster health endpoint..."
        HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

        echo "Cluster health test result:"
        echo "$HEALTH_TEST_RESULT"

        HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)
```

```
        if is_success_code "$HEALTH_HTTP_CODE"; then
            echo "# Cluster health authentication test successful! (HTTP
 $HEALTH_HTTP_CODE)"
            AUTH_SUCCESS=true
            AUTH_METHOD="basic"
        else
            echo "Cluster health authentication also failed with HTTP code:
 $HEALTH_HTTP_CODE"

            # Check for specific error patterns
            if echo "$AUTH_TEST_RESULT" | grep -q "anonymous is not authorized";
 then
                echo "Error: Request is being treated as anonymous
 (authentication not working)"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Unauthorized"; then
                echo "Error: Authentication credentials rejected"
            elif echo "$AUTH_TEST_RESULT" | grep -q "Forbidden"; then
                echo "Error: Authentication succeeded but access is forbidden"
            fi

            echo "Waiting additional time and retrying with exponential
 backoff..."

            # Retry authentication test with exponential backoff
            AUTH_RETRY_COUNT=0
            MAX_AUTH_RETRIES=5
            WAIT_TIME=60
            AUTH_SUCCESS=false

            while [[ $AUTH_RETRY_COUNT -lt $MAX_AUTH_RETRIES ]]; do
                echo "Retrying authentication test (attempt $((AUTH_RETRY_COUNT
+1))/$MAX_AUTH_RETRIES) after ${WAIT_TIME} seconds..."
                sleep $WAIT_TIME

                # Try both endpoints
                AUTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/" 2>&1)

                HTTP_CODE=$(echo "$AUTH_TEST_RESULT" | grep "HTTP_CODE:" | cut -
d: -f2)

                echo "Retry result (HTTP $HTTP_CODE):"
```

```
                echo "$AUTH_TEST_RESULT"

                if is_success_code "$HTTP_CODE"; then
                    echo "# Authentication test successful after retry! (HTTP
$HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                # Also try cluster health
                HEALTH_TEST_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
                    --netrc-file "$CREDENTIALS_FILE" \
                    --request GET \
                    "https://${DOMAIN_ENDPOINT}/_cluster/health" 2>&1)

                HEALTH_HTTP_CODE=$(echo "$HEALTH_TEST_RESULT" | grep "HTTP_CODE:"
| cut -d: -f2)

                if is_success_code "$HEALTH_HTTP_CODE"; then
                    echo "# Cluster health authentication successful after retry!
(HTTP $HEALTH_HTTP_CODE)"
                    AUTH_SUCCESS=true
                    AUTH_METHOD="basic"
                    break
                fi

                AUTH_RETRY_COUNT=$((AUTH_RETRY_COUNT+1))
                # Exponential backoff: double the wait time each retry (max 10
minutes)
                WAIT_TIME=$((WAIT_TIME * 2))
                if [[ $WAIT_TIME -gt 600 ]]; then
                    WAIT_TIME=600
                fi
            done
        fi
    fi

    # Proceed with data operations if authentication is working
    if [[ "$AUTH_SUCCESS" == "true" ]]; then
        echo "Authentication successful using $AUTH_METHOD method. Proceeding
with data operations."

        # Upload single document (matches tutorial exactly)
```

```
        echo "Uploading single document..."
        UPLOAD_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request PUT \
            --header 'Content-Type: application/json' \
            --data @"$SINGLE_MOVIE_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_doc/1" 2>&1)

        echo "Upload response:"
        echo "$UPLOAD_RESULT"

        UPLOAD_HTTP_CODE=$(echo "$UPLOAD_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        if is_success_code "$UPLOAD_HTTP_CODE" && echo "$UPLOAD_RESULT" | grep -q
 '"result"'; then
            echo "# Single document uploaded successfully! (HTTP
 $UPLOAD_HTTP_CODE)"
        else
            echo "# Warning: Single document upload may have failed (HTTP
 $UPLOAD_HTTP_CODE)"
        fi

        # Upload bulk documents (matches tutorial exactly)
        echo "Uploading bulk documents..."
        BULK_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request POST \
            --header 'Content-Type: application/x-ndjson' \
            --data-binary @"$BULK_MOVIES_FILE" \
            "https://${DOMAIN_ENDPOINT}/movies/_bulk" 2>&1)

        echo "Bulk upload response:"
        echo "$BULK_RESULT"

        BULK_HTTP_CODE=$(echo "$BULK_RESULT" | grep "HTTP_CODE:" | cut -d: -f2)
        if is_success_code "$BULK_HTTP_CODE" && echo "$BULK_RESULT" | grep -q
 '"errors": false'; then
            echo "# Bulk documents uploaded successfully! (HTTP $BULK_HTTP_CODE)"
        else
            echo "# Warning: Bulk document upload may have failed (HTTP
 $BULK_HTTP_CODE)"
        fi

        # Wait a moment for indexing
```

```
        echo "Waiting for documents to be indexed..."
        sleep 5

        # Step 3: Search Documents (matches tutorial exactly)
        echo "Searching for documents containing 'mars'..."
        SEARCH_RESULT=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true" 2>&1)

        SEARCH_HTTP_CODE=$(echo "$SEARCH_RESULT" | grep "HTTP_CODE:" | cut -d: -
f2)
        echo "Search results for 'mars' (HTTP $SEARCH_HTTP_CODE):"
        echo "$SEARCH_RESULT"

        echo "Searching for documents containing 'rebel'..."
        REBEL_SEARCH=$(curl -s -w "\nHTTP_CODE:%{http_code}" \
            --netrc-file "$CREDENTIALS_FILE" \
            --request GET \
            "https://${DOMAIN_ENDPOINT}/movies/_search?q=rebel&pretty=true" 2>&1)

        REBEL_HTTP_CODE=$(echo "$REBEL_SEARCH" | grep "HTTP_CODE:" | cut -d: -f2)
        echo "Search results for 'rebel' (HTTP $REBEL_HTTP_CODE):"
        echo "$REBEL_SEARCH"

        # Verify search results
        if is_success_code "$SEARCH_HTTP_CODE" && echo "$SEARCH_RESULT" | grep -q
 '"hits"'; then
            echo "# Search functionality is working!"
        else
            echo "# Warning: Search may not be working properly."
        fi

    else
        echo ""
        echo "========================================="
        echo "AUTHENTICATION TROUBLESHOOTING"
        echo "========================================="
        echo "Authentication failed after all retries. This may be due to:"
        echo "1. Fine-grained access control not fully initialized (most common)"
        echo "2. Domain configuration issues"
        echo "3. Network connectivity issues"
        echo "4. AWS credentials or permissions issues"
        echo ""
```

```
        echo "DOMAIN CONFIGURATION DEBUG:"
        echo "Let's check the domain configuration..."

        # Debug domain configuration
        DOMAIN_CONFIG=$(aws opensearch describe-domain --domain-
name "$DOMAIN_NAME" --query 'DomainStatus.{AdvancedSecurityOptions:
 AdvancedSecurityOptions, AccessPolicies: AccessPolicies}' --output json 2>&1)
        echo "Domain configuration:"
        echo "$DOMAIN_CONFIG"

        echo ""
        echo "MANUAL TESTING COMMANDS:"
        echo "You can try these commands manually in 10-15 minutes:"
        echo ""
        echo "# Test basic authentication:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/\""
        echo ""
        echo "# Test cluster health:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/_cluster/health\""
        echo ""
        echo "# Upload single document:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" --request PUT --
header 'Content-Type: application/json' --data @single_movie.json \"https://
${DOMAIN_ENDPOINT}/movies/_doc/1\""
        echo ""
        echo "# Search for documents:"
        echo "curl --user \"${MASTER_USER}:[PASSWORD]\" \"https://
${DOMAIN_ENDPOINT}/movies/_search?q=mars&pretty=true\""
        echo ""
        echo "TROUBLESHOOTING TIPS:"
        echo "- Wait 10-15 more minutes and try the manual commands"
        echo "- Check AWS CloudTrail logs for authentication errors"
        echo "- Verify your AWS region is correct: $AWS_REGION"
        echo "- Ensure your AWS credentials have OpenSearch permissions"
        echo "- Try accessing OpenSearch Dashboards to verify the master user
 works"
        echo ""
        echo "Skipping data upload and search operations for now."
        echo "The domain is created and accessible via OpenSearch Dashboards."
    fi

    # Securely remove credentials file
```

```
    shred -vfz -n 3 "$CREDENTIALS_FILE" 2>/dev/null || true
fi

# Display OpenSearch Dashboards URL (matches tutorial)
echo ""
echo "============================================="
echo "OPENSEARCH DASHBOARDS ACCESS"
echo "============================================="
echo "OpenSearch Dashboards URL: https://${DOMAIN_ENDPOINT}/_dashboards/"
echo "Username: $MASTER_USER"
echo "Password: [REDACTED - stored securely]"
echo ""
echo "You can access OpenSearch Dashboards using the credentials provided during
 domain creation."
echo "If you uploaded data successfully, you can create an index pattern for
 'movies'."
echo ""

# Summary of created resources
echo ""
echo "============================================="
echo "RESOURCES CREATED"
echo "============================================="
echo "OpenSearch Domain Name: $DOMAIN_NAME"
echo "OpenSearch Domain Endpoint: $DOMAIN_ENDPOINT"
echo "AWS Region: $AWS_REGION"
echo "Master Username: $MASTER_USER"
echo ""
echo "ESTIMATED COST: ~$0.038/hour (~$0.91/day) until deleted"
echo ""
echo "Make sure to save the domain name and endpoint for future reference."
echo ""

# Auto-confirm cleanup: Yes, clean up all resources
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Automatically cleaning up all created resources..."
CLEANUP_CHOICE="y"

if [[ "${CLEANUP_CHOICE,,}" == "y" ]]; then
    echo "Cleaning up resources..."
    if [ -n "${STACK_NAME:-}" ]; then
```

```
        echo "Deleting CloudFormation stack $STACK_NAME..."
        aws cloudformation delete-stack --stack-name "$STACK_NAME"
        echo "# Stack deletion initiated. This may take several minutes."
        echo "  Monitor: aws cloudformation describe-stacks --stack-name
 $STACK_NAME"
    else
        aws opensearch delete-domain --domain-name "$DOMAIN_NAME" 2>/dev/null
        echo "# Domain deletion initiated."
    fi
else
    echo "Resources will NOT be deleted automatically."
    echo ""
    echo "To delete later:"
    echo "  aws cloudformation delete-stack --stack-name ${STACK_NAME:-tutorial-
opensearch-XXXX}"
    echo ""
    echo "# IMPORTANT: Keeping these resources will incur ongoing AWS charges!"
fi

# Clean up temporary files (handled by trap)
echo "Cleaning up temporary files..."

# Disable the trap since we're handling cleanup manually
trap - EXIT

# Final cleanup
rm -rf "$TEMP_DIR" 2>/dev/null || true

echo ""
echo "=========================================="
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "=========================================="
echo "Script completed at $(date)"
echo "All output has been logged to: $LOG_FILE"
echo ""
echo "Next steps:"
echo "1. Access OpenSearch Dashboards at: https://${DOMAIN_ENDPOINT}/
_dashboards/"
echo "2. Create visualizations and dashboards"
echo "3. Explore the OpenSearch API"
echo "4. Remember to delete resources when done to avoid charges"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。

- CreateDomain
- DeleteDomain
- DescribeDomain
- GetCallerIdentity

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Amazon SageMaker Feature Store 入门

以下代码示例展示了如何：

- 设置 IAM 权限
- 创建 SageMaker 执行角色
- 创建特征组
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Amazon SageMaker Feature Store Tutorial Script - Version 3
# This script demonstrates how to use Amazon SageMaker Feature Store with AWS CLI

# Setup logging
LOG_FILE="sagemaker-featurestore-tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting SageMaker Feature Store tutorial script at $(date)"
echo "All commands and outputs will be logged to $LOG_FILE"
```

```
echo ""

# Track created resources for cleanup
CREATED_RESOURCES=()

# Function to handle errors
handle_error() {
    echo "ERROR: $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Function to check command status
check_status() {
    if echo "$1" | grep -i "error" > /dev/null; then
        handle_error "$1"
    fi
}

# Function to wait for feature group to be created
wait_for_feature_group() {
    local feature_group_name=$1
    local status="Creating"

    echo "Waiting for feature group ${feature_group_name} to be created..."

    while [ "$status" = "Creating" ]; do
        sleep 5
        status=$(aws sagemaker describe-feature-group \
            --feature-group-name "${feature_group_name}" \
            --query 'FeatureGroupStatus' \
            --output text)
        echo "Current status: ${status}"

        if [ "$status" = "Failed" ]; then
            handle_error "Feature group ${feature_group_name} creation failed"
        fi
    done

    echo "Feature group ${feature_group_name} is now ${status}"
}

# Function to clean up resources
```

```
cleanup_resources() {
    echo "Cleaning up resources..."

    # Clean up in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        resource="${CREATED_RESOURCES[$i]}"
        resource_type=$(echo "$resource" | cut -d: -f1)
        resource_name=$(echo "$resource" | cut -d: -f2)

        echo "Deleting $resource_type: $resource_name"

        case "$resource_type" in
            "FeatureGroup")
                aws sagemaker delete-feature-group --feature-group-name
 "$resource_name"
                ;;
            "S3Bucket")
                echo "Emptying S3 bucket: $resource_name"
                aws s3 rm "s3://$resource_name" --recursive 2>/dev/null
                echo "Deleting S3 bucket: $resource_name"
                aws s3api delete-bucket --bucket "$resource_name" 2>/dev/null
                ;;
            "IAMRole")
                echo "Detaching policies from role: $resource_name"
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>/dev/null
                aws iam detach-role-policy --role-name "$resource_name" --policy-
arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>/dev/null
                echo "Deleting IAM role: $resource_name"
                aws iam delete-role --role-name "$resource_name" 2>/dev/null
                ;;
            *)
                echo "Unknown resource type: $resource_type"
                ;;
        esac
    done
}

# Function to create SageMaker execution role
create_sagemaker_role() {
    local role_name="SageMakerFeatureStoreRole-$(openssl rand -hex 4)"

    echo "Creating SageMaker execution role: $role_name" >&2
```

```
    # Create trust policy document
    local trust_policy='{
        "Version":"2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "sagemaker.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }'

    # Create the role
    local role_result=$(aws iam create-role \
        --role-name "$role_name" \
        --assume-role-policy-document "$trust_policy" \
        --description "SageMaker execution role for Feature Store tutorial" 2>&1)

    if echo "$role_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to create IAM role: $role_result"
    fi

    echo "Role created successfully" >&2
    CREATED_RESOURCES+=("IAMRole:$role_name")

    # Tag the role
    echo "Tagging IAM role..." >&2
    aws iam tag-role --role-name "$role_name" --tags Key=project,Value=doc-smith
Key=tutorial,Value=sagemaker-featurestore 2>&1

    # Attach necessary policies
    echo "Attaching policies to role..." >&2

    # SageMaker execution policy
    local policy1_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonSageMakerFullAccess" 2>&1)

    if echo "$policy1_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach SageMaker policy: $policy1_result"
    fi
```

```
    # S3 access policy
    local policy2_result=$(aws iam attach-role-policy \
        --role-name "$role_name" \
        --policy-arn "arn:aws:iam::aws:policy/AmazonS3FullAccess" 2>&1)

    if echo "$policy2_result" | grep -i "error" > /dev/null; then
        handle_error "Failed to attach S3 policy: $policy2_result"
    fi

    # Get account ID for role ARN
    local account_id=$(aws sts get-caller-identity --query Account --output text)
    local role_arn="arn:aws:iam::${account_id}:role/${role_name}"

    echo "Role ARN: $role_arn" >&2
    echo "Waiting 10 seconds for role to propagate..." >&2
    sleep 10

    # Return only the role ARN to stdout
    echo "$role_arn"
}

# Handle SageMaker execution role
ROLE_ARN=""

if [ -z "$1" ]; then
    echo "Creating SageMaker execution role automatically..."
    ROLE_ARN=$(create_sagemaker_role)
    if [ -z "$ROLE_ARN" ]; then
        handle_error "Failed to create SageMaker execution role"
    fi
else
    ROLE_ARN="$1"

    # Validate the role ARN
    ROLE_NAME=$(echo "$ROLE_ARN" | sed 's/.*role\///')
    ROLE_CHECK=$(aws iam get-role --role-name "$ROLE_NAME" 2>&1)
    if echo "$ROLE_CHECK" | grep -i "error" > /dev/null; then
        echo "Creating a new role automatically..."
        ROLE_ARN=$(create_sagemaker_role)
        if [ -z "$ROLE_ARN" ]; then
            handle_error "Failed to create SageMaker execution role"
        fi
    fi
fi
```

```
# Handle cleanup option
AUTO_CLEANUP=""
if [ -n "$2" ]; then
    AUTO_CLEANUP="$2"
fi

# Generate a random identifier for resource names
RANDOM_ID=$(openssl rand -hex 4)
echo "Using random identifier: $RANDOM_ID"

# Set variables
ACCOUNT_ID=$(aws sts get-caller-identity --query Account --output text)
check_status "$ACCOUNT_ID"
echo "Account ID: $ACCOUNT_ID"

# Get current region
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"
    echo "No default region configured, using: $REGION"
else
    echo "Using region: $REGION"
fi
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="sagemaker-featurestore-${RANDOM_ID}-${ACCOUNT_ID}"
fi
PREFIX="featurestore-tutorial"
CURRENT_TIME=$(date +%s)

echo "Creating S3 bucket: $S3_BUCKET_NAME"
# Create bucket in current region (skip if using shared bucket)
if [ "$BUCKET_IS_SHARED" = "false" ]; then
if [ "$REGION" = "us-east-1" ]; then
```

```
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" 2>&1)
else
    BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" \
        --region "$REGION" \
        --create-bucket-configuration LocationConstraint="$REGION" 2>&1)
fi

if echo "$BUCKET_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to create S3 bucket: $BUCKET_RESULT"
    exit 1
fi

echo "$BUCKET_RESULT"
CREATED_RESOURCES+=("S3Bucket:$S3_BUCKET_NAME")

# Tag the S3 bucket
echo "Tagging S3 bucket: $S3_BUCKET_NAME"
aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=sagemaker-
featurestore}]' 2>&1

# Block public access to the bucket
BLOCK_RESULT=$(aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
 2>&1)

if echo "$BLOCK_RESULT" | grep -i "error" > /dev/null; then
    echo "Failed to block public access to S3 bucket: $BLOCK_RESULT"
    cleanup_resources
    exit 1
fi
else
    echo "Using shared bucket (skipping creation)"
fi

# Create feature groups
echo "Creating feature groups..."

# Create customers feature group
CUSTOMERS_FEATURE_GROUP_NAME="customers-feature-group-${RANDOM_ID}"
echo "Creating customers feature group: $CUSTOMERS_FEATURE_GROUP_NAME"
```

```
CUSTOMERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "name", "FeatureType": "String"},
        {"FeatureName": "age", "FeatureType": "Integral"},
        {"FeatureName": "address", "FeatureType": "String"},
        {"FeatureName": "membership_type", "FeatureType": "String"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$CUSTOMERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create customers feature group: $CUSTOMERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$CUSTOMERS_FEATURE_GROUP_NAME")

# Create orders feature group
ORDERS_FEATURE_GROUP_NAME="orders-feature-group-${RANDOM_ID}"
echo "Creating orders feature group: $ORDERS_FEATURE_GROUP_NAME"

ORDERS_RESPONSE=$(aws sagemaker create-feature-group \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record-identifier-feature-name "customer_id" \
    --event-time-feature-name "EventTime" \
    --feature-definitions '[
        {"FeatureName": "customer_id", "FeatureType": "Integral"},
        {"FeatureName": "order_id", "FeatureType": "String"},
```

```
        {"FeatureName": "order_date", "FeatureType": "String"},
        {"FeatureName": "product", "FeatureType": "String"},
        {"FeatureName": "quantity", "FeatureType": "Integral"},
        {"FeatureName": "amount", "FeatureType": "Fractional"},
        {"FeatureName": "EventTime", "FeatureType": "Fractional"}
    ]' \
    --online-store-config '{"EnableOnlineStore": true}' \
    --offline-store-config '{
        "S3StorageConfig": {
            "S3Uri": "s3://'${S3_BUCKET_NAME}'/'${PREFIX}'"
        },
        "DisableGlueTableCreation": false
    }' \
    --role-arn "$ROLE_ARN" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=sagemaker-featurestore
 2>&1)

if echo "$ORDERS_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to create orders feature group: $ORDERS_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDERS_RESPONSE"
CREATED_RESOURCES+=("FeatureGroup:$ORDERS_FEATURE_GROUP_NAME")

# Wait for feature groups to be created
wait_for_feature_group "$CUSTOMERS_FEATURE_GROUP_NAME"
wait_for_feature_group "$ORDERS_FEATURE_GROUP_NAME"

# Ingest data into feature groups
echo "Ingesting data into feature groups..."

# Ingest customer data
echo "Ingesting customer data..."
CUSTOMER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "name", "ValueAsString": "John Doe"},
        {"FeatureName": "age", "ValueAsString": "35"},
        {"FeatureName": "address", "ValueAsString": "123 Main St"},
        {"FeatureName": "membership_type", "ValueAsString": "premium"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
```

```
    ]' 2>&1)

if echo "$CUSTOMER1_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 1 data: $CUSTOMER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER1_RESPONSE"

CUSTOMER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "name", "ValueAsString": "Jane Smith"},
        {"FeatureName": "age", "ValueAsString": "28"},
        {"FeatureName": "address", "ValueAsString": "456 Oak Ave"},
        {"FeatureName": "membership_type", "ValueAsString": "standard"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$CUSTOMER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest customer 2 data: $CUSTOMER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER2_RESPONSE"

# Ingest order data
echo "Ingesting order data..."
ORDER1_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "573291"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-001"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-15"},
        {"FeatureName": "product", "ValueAsString": "Laptop"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "1299.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER1_RESPONSE" | grep -i "error" > /dev/null; then
```

```
    echo "Failed to ingest order 1 data: $ORDER1_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER1_RESPONSE"

ORDER2_RESPONSE=$(aws sagemaker-featurestore-runtime put-record \
    --feature-group-name "$ORDERS_FEATURE_GROUP_NAME" \
    --record '[
        {"FeatureName": "customer_id", "ValueAsString": "109382"},
        {"FeatureName": "order_id", "ValueAsString": "ORD-002"},
        {"FeatureName": "order_date", "ValueAsString": "2023-01-20"},
        {"FeatureName": "product", "ValueAsString": "Smartphone"},
        {"FeatureName": "quantity", "ValueAsString": "1"},
        {"FeatureName": "amount", "ValueAsString": "899.99"},
        {"FeatureName": "EventTime", "ValueAsString": "'${CURRENT_TIME}'"}
    ]' 2>&1)

if echo "$ORDER2_RESPONSE" | grep -i "error" > /dev/null; then
    echo "Failed to ingest order 2 data: $ORDER2_RESPONSE"
    cleanup_resources
    exit 1
fi

echo "$ORDER2_RESPONSE"

# Retrieve records from feature groups
echo "Retrieving records from feature groups..."

# Get a single customer record
echo "Getting customer record with ID 573291:"
CUSTOMER_RECORD=$(aws sagemaker-featurestore-runtime get-record \
    --feature-group-name "$CUSTOMERS_FEATURE_GROUP_NAME" \
    --record-identifier-value-as-string "573291" 2>&1)

if echo "$CUSTOMER_RECORD" | grep -i "error" > /dev/null; then
    echo "Failed to get customer record: $CUSTOMER_RECORD"
    cleanup_resources
    exit 1
fi

echo "$CUSTOMER_RECORD"
```

```
# Get multiple records using batch-get-record
echo "Getting multiple records using batch-get-record:"
BATCH_RECORDS=$(aws sagemaker-featurestore-runtime batch-get-record \
    --identifiers '[
        {
            "FeatureGroupName": "'${CUSTOMERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        },
        {
            "FeatureGroupName": "'${ORDERS_FEATURE_GROUP_NAME}'",
            "RecordIdentifiersValueAsString": ["573291", "109382"]
        }
    ]' 2>&1)

if echo "$BATCH_RECORDS" | grep -i "error" > /dev/null && ! echo "$BATCH_RECORDS"
 | grep -i "Records" > /dev/null; then
    echo "Failed to get batch records: $BATCH_RECORDS"
    cleanup_resources
    exit 1
fi

echo "$BATCH_RECORDS"

# List feature groups
echo "Listing feature groups:"
FEATURE_GROUPS=$(aws sagemaker list-feature-groups 2>&1)

if echo "$FEATURE_GROUPS" | grep -i "error" > /dev/null; then
    echo "Failed to list feature groups: $FEATURE_GROUPS"
    cleanup_resources
    exit 1
fi

echo "$FEATURE_GROUPS"

# Display summary of created resources
echo ""
echo "============================================="
echo "TUTORIAL COMPLETED SUCCESSFULLY!"
echo "============================================="
echo "Resources created:"
echo "- S3 Bucket: $S3_BUCKET_NAME"
echo "- Customers Feature Group: $CUSTOMERS_FEATURE_GROUP_NAME"
echo "- Orders Feature Group: $ORDERS_FEATURE_GROUP_NAME"
```

```
if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
    echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o 'IAMRole:
[^[:space:]]*' | cut -d: -f2)"
fi
echo ""
echo "You can now:"
echo "1. View your feature groups in the SageMaker console"
echo "2. Query the offline store using Amazon Athena"
echo "3. Use the feature groups in your ML workflows"
echo "============================================="
echo ""

# Handle cleanup
if [ "$AUTO_CLEANUP" = "y" ]; then
    echo "Auto-cleanup enabled. Starting cleanup..."
    cleanup_resources
    echo "Cleanup completed."
elif [ "$AUTO_CLEANUP" = "n" ]; then
    echo "Auto-cleanup disabled. Resources will remain in your account."
    echo "To clean up later, run this script again with cleanup option 'y'"
else
    echo "============================================="
    echo "CLEANUP CONFIRMATION"
    echo "============================================="
    echo "Do you want to clean up all created resources? (y/n): "
    read -r CLEANUP_CHOICE

    if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
        echo "Starting cleanup..."
        cleanup_resources
        echo "Cleanup completed."
    else
        echo "Skipping cleanup. Resources will remain in your account."
        echo "To clean up later, delete the following resources:"
        echo "- Feature Groups: $CUSTOMERS_FEATURE_GROUP_NAME,
 $ORDERS_FEATURE_GROUP_NAME"
        echo "- S3 Bucket: $S3_BUCKET_NAME"
        if [[ " ${CREATED_RESOURCES[@]} " =~ " IAMRole:" ]]; then
            echo "- IAM Role: $(echo "${CREATED_RESOURCES[@]}" | grep -o
 'IAMRole:[^[:space:]]*' | cut -d: -f2)"
        fi
        echo ""
        echo "Estimated ongoing cost: ~$0.01 per month for online store"
    fi
```

```
fi

echo "Script completed at $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateBucket
  - CreateFeatureGroup
  - CreateRole
  - DeleteBucket
  - DeleteFeatureGroup
  - DeleteRole
  - DescribeFeatureGroup
  - DetachRolePolicy
  - GetCallerIdentity
  - GetRole
  - ListFeatureGroups
  - PutPublicAccessBlock
  - Rm

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

Config 入门

以下代码示例展示了如何：

- 创建 Amazon S3 存储桶
- 创建 Amazon SNS 主题
- 为 Config 创建 IAM 角色
- 设置 Config 配置记录器
- 设置 Config 传输通道
- 启动配置记录器
- 验证 Config 设置

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS Config Setup Script (v2)
# This script sets up AWS Config with the AWS CLI

# Error handling
set -e
LOGFILE="aws-config-setup-v2.log"
touch $LOGFILE
exec > >(tee -a $LOGFILE)
exec 2>&1

# Function to handle errors
handle_error() {
    echo "ERROR: An error occurred at line $1"
    echo "Attempting to clean up resources..."
    cleanup_resources
    exit 1
}

# Set trap for error handling
trap 'handle_error $LINENO' ERR

# Function to generate random identifier
generate_random_id() {
    echo $(openssl rand -hex 6)
}

# Function to check if command was successful
check_command() {
    if echo "$1" | grep -i "error" > /dev/null; then
        echo "ERROR: $1"
```

```
        return 1
    fi
    return 0
}

# Function to clean up resources
cleanup_resources() {
    if [ -n "$CONFIG_RECORDER_NAME" ]; then
        echo "Stopping configuration recorder..."
        aws configservice stop-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    # Check if we created a new delivery channel before trying to delete it
    if [ -n "$DELIVERY_CHANNEL_NAME" ] && [ "$CREATED_NEW_DELIVERY_CHANNEL" =
 "true" ]; then
        echo "Deleting delivery channel..."
        aws configservice delete-delivery-channel --delivery-channel-name
 "$DELIVERY_CHANNEL_NAME" 2>/dev/null || true
    fi

    if [ -n "$CONFIG_RECORDER_NAME" ] && [ "$CREATED_NEW_CONFIG_RECORDER" =
 "true" ]; then
        echo "Deleting configuration recorder..."
        aws configservice delete-configuration-recorder --configuration-recorder-
name "$CONFIG_RECORDER_NAME" 2>/dev/null || true
    fi

    if [ -n "$ROLE_NAME" ]; then
        if [ -n "$POLICY_NAME" ]; then
            echo "Detaching custom policy from role..."
            aws iam delete-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" 2>/dev/null || true
        fi

        if [ -n "$MANAGED_POLICY_ARN" ]; then
            echo "Detaching managed policy from role..."
            aws iam detach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN" 2>/dev/null || true
        fi

        echo "Deleting IAM role..."
        aws iam delete-role --role-name "$ROLE_NAME" 2>/dev/null || true
    fi
```

```
    if [ -n "$SNS_TOPIC_ARN" ]; then
        echo "Deleting SNS topic..."
        aws sns delete-topic --topic-arn "$SNS_TOPIC_ARN" 2>/dev/null || true
    fi

    if [ -n "$S3_BUCKET_NAME" ]; then
        echo "Emptying S3 bucket..."
        aws s3 rm "s3://$S3_BUCKET_NAME" --recursive 2>/dev/null || true

        echo "Deleting S3 bucket..."
        if [ "$BUCKET_IS_SHARED" = "false" ]; then
            aws s3api delete-bucket --bucket "$S3_BUCKET_NAME" 2>/dev/null ||
 true
        fi
    fi
}

# Function to display created resources
display_resources() {
    echo ""
    echo "========================================="
    echo "CREATED RESOURCES"
    echo "========================================="
    echo "S3 Bucket: $S3_BUCKET_NAME"
    echo "SNS Topic ARN: $SNS_TOPIC_ARN"
    echo "IAM Role: $ROLE_NAME"
    if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (newly created)"
    else
        echo "Configuration Recorder: $CONFIG_RECORDER_NAME (existing)"
    fi
    if [ "$CREATED_NEW_DELIVERY_CHANNEL" = "true" ]; then
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (newly created)"
    else
        echo "Delivery Channel: $DELIVERY_CHANNEL_NAME (existing)"
    fi
    echo "========================================="
}

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
if [ -z "$ACCOUNT_ID" ]; then
```

```
    echo "ERROR: Failed to get AWS account ID"
    exit 1
fi
echo "AWS Account ID: $ACCOUNT_ID"

# Generate random identifier for resources
RANDOM_ID=$(generate_random_id)
echo "Generated random identifier: $RANDOM_ID"

# Step 1: Create an S3 bucket
# Check for shared prereq bucket
PREREQ_BUCKET=$(aws cloudformation describe-stacks --stack-name tutorial-prereqs-
bucket \
    --query 'Stacks[0].Outputs[?OutputKey==`BucketName`].OutputValue' --output
 text 2>/dev/null)
if [ -n "$PREREQ_BUCKET" ] && [ "$PREREQ_BUCKET" != "None" ]; then
    S3_BUCKET_NAME="$PREREQ_BUCKET"
    BUCKET_IS_SHARED=true
    echo "Using shared bucket: $S3_BUCKET_NAME"
else
    BUCKET_IS_SHARED=false
    S3_BUCKET_NAME="configservice-${RANDOM_ID}"
    echo "Creating S3 bucket: $S3_BUCKET_NAME"
fi

# Get the current region
AWS_REGION=$(aws configure get region)
if [ -z "$AWS_REGION" ]; then
    AWS_REGION="us-east-1"  # Default to us-east-1 if no region is configured
fi
echo "Using AWS Region: $AWS_REGION"

# Create bucket with appropriate command based on region
if [ "$BUCKET_IS_SHARED" = "false" ]; then
    if [ "$AWS_REGION" = "us-east-1" ]; then
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME")
    else
        BUCKET_RESULT=$(aws s3api create-bucket --bucket "$S3_BUCKET_NAME" --
create-bucket-configuration LocationConstraint="$AWS_REGION")
    fi
    check_command "$BUCKET_RESULT"
    echo "S3 bucket created: $S3_BUCKET_NAME"
```

```
    aws s3api put-bucket-tagging --bucket "$S3_BUCKET_NAME" --tagging
 'TagSet=[{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-config-gs}]'
    echo "Tags applied to S3 bucket"
else
    echo "Using shared bucket: $S3_BUCKET_NAME (skipping creation)"
fi

# Block public access for the bucket
aws s3api put-public-access-block \
    --bucket "$S3_BUCKET_NAME" \
    --public-access-block-configuration
 "BlockPublicAcls=true,IgnorePublicAcls=true,BlockPublicPolicy=true,RestrictPublicBuckets
echo "Public access blocked for bucket"

# Step 2: Create an SNS topic
TOPIC_NAME="config-topic-${RANDOM_ID}"
echo "Creating SNS topic: $TOPIC_NAME"
SNS_RESULT=$(aws sns create-topic --name "$TOPIC_NAME" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs)
check_command "$SNS_RESULT"
SNS_TOPIC_ARN=$(echo "$SNS_RESULT" | grep -o 'arn:aws:sns:[^"]*')
echo "SNS topic created: $SNS_TOPIC_ARN"

# Step 3: Create an IAM role for AWS Config
ROLE_NAME="config-role-${RANDOM_ID}"
POLICY_NAME="config-delivery-permissions"
MANAGED_POLICY_ARN="arn:aws:iam::aws:policy/service-role/AWS_ConfigRole"

echo "Creating trust policy document..."
cat > config-trust-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "config.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF
```

```
echo "Creating IAM role: $ROLE_NAME"
ROLE_RESULT=$(aws iam create-role --role-name "$ROLE_NAME" --assume-role-policy-
document file://config-trust-policy.json)
check_command "$ROLE_RESULT"
ROLE_ARN=$(echo "$ROLE_RESULT" | grep -o 'arn:aws:iam::[^"]*' | head -1)
echo "IAM role created: $ROLE_ARN"

aws iam tag-role --role-name "$ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-config-gs
echo "Tags applied to IAM role"

echo "Attaching AWS managed policy to role..."
ATTACH_RESULT=$(aws iam attach-role-policy --role-name "$ROLE_NAME" --policy-arn
 "$MANAGED_POLICY_ARN")
check_command "$ATTACH_RESULT"
echo "AWS managed policy attached"

echo "Creating custom policy document for S3 and SNS access..."
cat > config-delivery-permissions.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}/AWSLogs/${ACCOUNT_ID}/*",
      "Condition": {
        "StringLike": {
          "s3:x-amz-acl": "bucket-owner-full-control"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetBucketAcl"
      ],
      "Resource": "arn:aws:s3:::${S3_BUCKET_NAME}"
    },
    {
      "Effect": "Allow",
      "Action": [
```

```
        "sns:Publish"
      ],
      "Resource": "${SNS_TOPIC_ARN}"
    }
  ]
}
EOF

echo "Attaching custom policy to role..."
POLICY_RESULT=$(aws iam put-role-policy --role-name "$ROLE_NAME" --policy-name
 "$POLICY_NAME" --policy-document file://config-delivery-permissions.json)
check_command "$POLICY_RESULT"
echo "Custom policy attached"

# Wait for IAM role to propagate
echo "Waiting for IAM role to propagate (15 seconds)..."
sleep 15

# Step 4: Check if configuration recorder already exists
CONFIG_RECORDER_NAME="default"
CREATED_NEW_CONFIG_RECORDER="false"

echo "Checking for existing configuration recorder..."
EXISTING_RECORDERS=$(aws configservice describe-configuration-recorders 2>/dev/
null || echo "")
if echo "$EXISTING_RECORDERS" | grep -q "name"; then
    echo "Configuration recorder already exists. Will update it."
    # Get the name of the existing recorder
    CONFIG_RECORDER_NAME=$(echo "$EXISTING_RECORDERS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing configuration recorder: $CONFIG_RECORDER_NAME"
else
    echo "No existing configuration recorder found. Will create a new one."
    CREATED_NEW_CONFIG_RECORDER="true"
fi

echo "Creating configuration recorder configuration..."
cat > configurationRecorder.json << EOF
{
  "name": "${CONFIG_RECORDER_NAME}",
  "roleARN": "${ROLE_ARN}",
  "recordingMode": {
    "recordingFrequency": "CONTINUOUS"
  }
```

```
}
EOF

echo "Creating recording group configuration..."
cat > recordingGroup.json << EOF
{
  "allSupported": true,
  "includeGlobalResourceTypes": true
}
EOF

echo "Setting up configuration recorder..."
RECORDER_RESULT=$(aws configservice put-configuration-recorder --configuration-
recorder file://configurationRecorder.json --recording-group file://
recordingGroup.json)
check_command "$RECORDER_RESULT"
echo "Configuration recorder set up"

if [ "$CREATED_NEW_CONFIG_RECORDER" = "true" ]; then
    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:config-recorder/${CONFIG_RECORDER_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to configuration recorder"
fi

# Step 5: Check if delivery channel already exists
DELIVERY_CHANNEL_NAME="default"
CREATED_NEW_DELIVERY_CHANNEL="false"

echo "Checking for existing delivery channel..."
EXISTING_CHANNELS=$(aws configservice describe-delivery-channels 2>/dev/null ||
 echo "")
if echo "$EXISTING_CHANNELS" | grep -q "name"; then
    echo "Delivery channel already exists."
    # Get the name of the existing channel
    DELIVERY_CHANNEL_NAME=$(echo "$EXISTING_CHANNELS" | grep -o '"name": "[^"]*"'
 | head -1 | cut -d'"' -f4)
    echo "Using existing delivery channel: $DELIVERY_CHANNEL_NAME"

    # Update the existing delivery channel
    echo "Creating delivery channel configuration for update..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
```

```
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Updating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel updated"
else
    echo "No existing delivery channel found. Will create a new one."
    CREATED_NEW_DELIVERY_CHANNEL="true"

    echo "Creating delivery channel configuration..."
    cat > deliveryChannel.json << EOF
{
  "name": "${DELIVERY_CHANNEL_NAME}",
  "s3BucketName": "${S3_BUCKET_NAME}",
  "snsTopicARN": "${SNS_TOPIC_ARN}",
  "configSnapshotDeliveryProperties": {
    "deliveryFrequency": "Six_Hours"
  }
}
EOF

    echo "Creating delivery channel..."
    CHANNEL_RESULT=$(aws configservice put-delivery-channel --delivery-channel
 file://deliveryChannel.json)
    check_command "$CHANNEL_RESULT"
    echo "Delivery channel created"

    aws configservice tag-resource --resource-arn "arn:aws:config:
${AWS_REGION}:${ACCOUNT_ID}:delivery-channel/${DELIVERY_CHANNEL_NAME}" --tags
 Key=project,Value=doc-smith Key=tutorial,Value=aws-config-gs
    echo "Tags applied to delivery channel"
fi

# Step 6: Start the configuration recorder
echo "Checking configuration recorder status..."
```

```
RECORDER_STATUS=$(aws configservice describe-configuration-recorder-status 2>/
dev/null || echo "")
if echo "$RECORDER_STATUS" | grep -q '"recording": true'; then
    echo "Configuration recorder is already running."
else
    echo "Starting configuration recorder..."
    START_RESULT=$(aws configservice start-configuration-recorder --
configuration-recorder-name "$CONFIG_RECORDER_NAME")
    check_command "$START_RESULT"
    echo "Configuration recorder started"
fi

# Step 7: Verify the AWS Config setup
echo "Verifying delivery channel..."
VERIFY_CHANNEL=$(aws configservice describe-delivery-channels)
check_command "$VERIFY_CHANNEL"
echo "$VERIFY_CHANNEL"

echo "Verifying configuration recorder..."
VERIFY_RECORDER=$(aws configservice describe-configuration-recorders)
check_command "$VERIFY_RECORDER"
echo "$VERIFY_RECORDER"

echo "Verifying configuration recorder status..."
VERIFY_STATUS=$(aws configservice describe-configuration-recorder-status)
check_command "$VERIFY_STATUS"
echo "$VERIFY_STATUS"

# Display created resources
display_resources

# Ask if user wants to clean up resources
echo ""
echo "==========================================="
echo "CLEANUP CONFIRMATION"
echo "==========================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE='y'

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..."
    cleanup_resources
    echo "Cleanup completed."
else
```

```
    echo "Resources will not be cleaned up. You can manually clean them up
 later."
fi

echo "Script completed successfully!"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateBucket
  - CreateRole
  - CreateTopic
  - DeleteBucket
  - DeleteConfigurationRecorder
  - DeleteDeliveryChannel
  - DeleteRole
  - DeleteRolePolicy
  - DeleteTopic
  - DescribeConfigurationRecorderStatus
  - DescribeConfigurationRecorders
  - DescribeDeliveryChannels
  - DetachRolePolicy
  - GetCallerIdentity
  - PutConfigurationRecorder
  - PutDeliveryChannel
  - PutPublicAccessBlock
  - PutRolePolicy
  - Rm
  - StartConfigurationRecorder
  - StopConfigurationRecorder

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

终端用户消息发送推送服务入门

以下代码示例展示了如何：

- 创建 应用程序
- 启用推送通知通道
- 发送推送通知
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS End User Messaging Push Getting Started Script
# This script creates an AWS End User Messaging Push application and demonstrates
# how to enable push notification channels and send a test message.
#
# Prerequisites:
# - AWS CLI installed and configured
# - Appropriate IAM permissions for Pinpoint operations
#
# Usage: ./aws-end-user-messaging-gs.sh [--auto-cleanup]

set -uo pipefail

# Security: Set secure umask for created files
umask 0077

# Set up logging
LOG_DIR="./aws-eump-logs"
mkdir -p "$LOG_DIR"
LOG_FILE="$LOG_DIR/aws-end-user-messaging-push-script-$(date +%Y%m%d-%H%M%S).log"
```

```
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS End User Messaging Push setup script..."
echo "Logging to $LOG_FILE"
echo "Timestamp: $(date)"

# Security: Track created resources for cleanup
declare -a TEMP_FILES=()
declare -a AWS_RESOURCES=()

# Cleanup function with improved security
cleanup() {
    local exit_code=$?
    echo "Cleaning up temporary resources..."

    # Remove temporary files securely
    for temp_file in "${TEMP_FILES[@]+"${TEMP_FILES[@]}"}"; do
        if [ -f "$temp_file" ]; then
            shred -vfz -n 3 "$temp_file" 2>/dev/null || rm -f "$temp_file"
        fi
    done

    # Optionally delete AWS resources
    if [ "${DELETE_AWS_RESOURCES:-false}" = "true" ]; then
        for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
            echo "Deleting AWS resource: $resource"
            aws pinpoint delete-app --application-id "$resource" 2>/dev/null || \
                echo "Warning: Failed to delete application $resource"
        done
    fi

    exit "$exit_code"
}

trap cleanup EXIT INT TERM

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2
    local ignore_error=${3:-false}

    if echo "$output" | grep -qi "error\|exception\|fail"; then
        echo "ERROR: Command failed: $cmd" >&2
```

```
        echo "Error details: $output" >&2

        if [ "$ignore_error" = "true" ]; then
            echo "Ignoring error and continuing..." >&2
            return 1
        else
            return 2
        fi
    fi

    return 0
}

# Function to validate AWS CLI is configured
validate_aws_cli() {
    echo "Validating AWS CLI configuration..."

    # Check if AWS CLI is installed
    if ! command -v aws &> /dev/null; then
        echo "ERROR: AWS CLI is not installed. Please install it first." >&2
        echo "Visit: https://docs.aws.amazon.com/cli/latest/userguide/getting-
started-install.html" >&2
        return 1
    fi

    # Check AWS CLI version
    AWS_VERSION=$(aws --version 2>&1 | head -n1)
    echo "AWS CLI version: $AWS_VERSION"

    # Verify credentials are set (check for credential env vars or config file)
    if ! aws sts get-caller-identity &> /dev/null; then
        echo "ERROR: AWS CLI credentials are not configured or invalid." >&2
        echo "Please configure credentials via environment variables, credential
 file, or 'aws configure'" >&2
        return 1
    fi

    # Get current AWS identity and region
    CALLER_IDENTITY=$(aws sts get-caller-identity)
    CURRENT_REGION=$(aws configure get region 2>/dev/null || echo "us-east-1")
    echo "AWS CLI configured for:"
    echo "$CALLER_IDENTITY"
    echo "Current region: $CURRENT_REGION"
    echo ""
```

```
    return 0
}

# Function to check if jq is available for JSON parsing
check_json_tools() {
    if command -v jq &> /dev/null; then
        USE_JQ=true
        echo "jq is available for JSON parsing"
    else
        USE_JQ=false
        echo "jq is not available, using grep for JSON parsing"
        echo "Consider installing jq for better JSON handling: https://
stedolan.github.io/jq/"
    fi
}

# Function to extract JSON values safely
extract_json_value() {
    local json=$1
    local key=$2

    if [ "$USE_JQ" = "true" ]; then
        echo "$json" | jq -r ".ApplicationResponse.$key // empty" 2>/dev/null ||
 echo ""
    else
        # Fallback to grep method with better validation
        echo "$json" | grep -o "\"$key\": \"[^\"]*" | cut -d'"' -f4 | head -n1 ||
 echo ""
    fi
}

# Function to validate required IAM permissions
validate_permissions() {
    echo "Validating IAM permissions..."

    # Test basic Pinpoint permissions
    if ! aws pinpoint get-apps > /dev/null 2>&1; then
        echo "WARNING: Unable to list Pinpoint applications." >&2
        echo "Please ensure you have appropriate IAM permissions for Pinpoint
 operations." >&2
        echo "Required permissions:" >&2
        echo "  - mobiletargeting:GetApps" >&2
        echo "  - mobiletargeting:CreateApp" >&2
```

```
        echo "  - mobiletargeting:DeleteApp" >&2
        echo "  - mobiletargeting:UpdateGcmChannel" >&2
        echo "  - mobiletargeting:UpdateApnsChannel" >&2
        echo "  - mobiletargeting:SendMessages" >&2
    else
        echo "Basic Pinpoint permissions validated."
    fi
}

# Function to validate input parameters
validate_input() {
    local app_name=$1

    # Validate app name length and characters
    if [ ${#app_name} -gt 64 ]; then
        echo "ERROR: Application name exceeds maximum length of 64 characters"
 >&2
        return 1
    fi

    if ! [[ "$app_name" =~ ^[a-zA-Z0-9_-]+$ ]]; then
        echo "ERROR: Application name contains invalid characters" >&2
        return 1
    fi

    return 0
}

# Function to create secure temporary files
create_temp_file() {
    local temp_file
    temp_file=$(mktemp) || {
        echo "ERROR: Failed to create temporary file" >&2
        return 1
    }
    chmod 600 "$temp_file"
    TEMP_FILES+=("$temp_file")
    echo "$temp_file"
}

# Validate prerequisites
if ! validate_aws_cli; then
    exit 1
fi
```

```
check_json_tools
validate_permissions

# Generate a random suffix for resource names to avoid conflicts
RANDOM_SUFFIX=$(LC_ALL=C tr -dc 'a-z0-9' < /dev/urandom | fold -w 8 | head -n1)
APP_NAME="PushNotificationApp-${RANDOM_SUFFIX}"

# Validate input
if ! validate_input "$APP_NAME"; then
    exit 1
fi

echo "Creating application with name: $APP_NAME"

# Step 1: Create an application
echo "Executing: aws pinpoint create-app --create-application-request Name=
${APP_NAME}"
CREATE_APP_OUTPUT=$(aws pinpoint create-app --create-application-request "Name=
${APP_NAME},tags={project=doc-smith,tutorial=aws-end-user-messaging-gs}" 2>&1)

if ! check_error "$CREATE_APP_OUTPUT" "create-app"; then
    exit 1
fi

echo "Application created successfully:"
echo "$CREATE_APP_OUTPUT"

# Extract the application ID from the output
APP_ID=$(extract_json_value "$CREATE_APP_OUTPUT" "Id")

if [ -z "$APP_ID" ] || [ "$APP_ID" = "null" ]; then
    echo "ERROR: Failed to extract application ID from output" >&2
    echo "Output was: $CREATE_APP_OUTPUT" >&2
    exit 1
fi

echo "Application ID: $APP_ID"
AWS_RESOURCES+=("$APP_ID")

# Step 2: Enable FCM (GCM) channel with a sample API key
echo ""
echo "========================================="
echo "ENABLING FCM (GCM) CHANNEL"
```

```
echo "============================================="
echo "Note: This is using a placeholder API key for demonstration purposes only."
echo "In a production environment, you should use your actual FCM API key from
 Firebase Console."
echo ""
echo "IMPORTANT: The following command will likely fail because we're using a
 placeholder API key."
echo "This is expected behavior for this demonstration script."

echo "Executing: aws pinpoint update-gcm-channel --application-id $APP_ID --gcm-
channel-request ..."
UPDATE_GCM_OUTPUT=$(aws pinpoint update-gcm-channel \
    --application-id "$APP_ID" \
    --gcm-channel-request '{"Enabled": true, "ApiKey": "sample-fcm-api-key-for-
demo-only"}' 2>&1)

# We'll ignore this specific error since we're using a placeholder API key
if check_error "$UPDATE_GCM_OUTPUT" "update-gcm-channel" "true"; then
    echo "FCM channel enabled successfully:"
    echo "$UPDATE_GCM_OUTPUT"
else
    echo "As expected, FCM channel update failed with the placeholder API key."
    echo "Error details: $UPDATE_GCM_OUTPUT"
    echo ""
    echo "To enable FCM in production:"
    echo "1. Go to Firebase Console (https://console.firebase.google.com/)"
    echo "2. Create or select your project"
    echo "3. Go to Project Settings > Cloud Messaging"
    echo "4. Copy the Server Key"
    echo "5. Replace 'sample-fcm-api-key-for-demo-only' with your actual Server
 Key"
fi

# Step 3: Try to enable APNS channel (this will also fail without real
 certificates)
echo ""
echo "============================================="
echo "ENABLING APNS CHANNEL (OPTIONAL)"
echo "============================================="
echo "Attempting to enable APNS channel with placeholder certificate..."
echo "This will also fail without real APNS certificates, which is expected."

# Create a placeholder APNS configuration
```

```
echo "Executing: aws pinpoint update-apns-channel --application-id $APP_ID --
apns-channel-request ..."
UPDATE_APNS_OUTPUT=$(aws pinpoint update-apns-channel \
    --application-id "$APP_ID" \
    --apns-channel-request '{"Enabled": true, "Certificate": "placeholder-
certificate", "PrivateKey": "placeholder-private-key"}' 2>&1)

if check_error "$UPDATE_APNS_OUTPUT" "update-apns-channel" "true"; then
    echo "APNS channel enabled successfully:"
    echo "$UPDATE_APNS_OUTPUT"
else
    echo "As expected, APNS channel update failed with placeholder certificates."
    echo "Error details: $UPDATE_APNS_OUTPUT"
    echo ""
    echo "To enable APNS in production:"
    echo "1. Generate APNS certificates from Apple Developer Console"
    echo "2. Convert certificates to PEM format"
    echo "3. Use the actual certificate and private key in the update-apns-
channel command"
fi

# Step 4: Create message files for different platforms
echo ""
echo "=========================================="
echo "CREATING MESSAGE FILES"
echo "=========================================="

# Create FCM message file securely
GCM_MESSAGE_FILE=$(create_temp_file)
echo "Creating FCM message file..."
cat > "$GCM_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-FCM": {
      "ChannelType": "GCM"
    }
  },
  "MessageConfiguration": {
    "GCMMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an FCM
 notification.",
      "Priority": "normal",
      "SilentPush": false,
```

```
      "Title": "My First FCM Push Notification",
      "TimeToLive": 30,
      "Data": {
        "key1": "value1",
        "key2": "value2"
      }
    }
  }
}
EOF

# Create APNS message file securely
APNS_MESSAGE_FILE=$(create_temp_file)
echo "Creating APNS message file..."
cat > "$APNS_MESSAGE_FILE" << 'EOF'
{
  "Addresses": {
    "SAMPLE-DEVICE-TOKEN-APNS": {
      "ChannelType": "APNS"
    }
  },
  "MessageConfiguration": {
    "APNSMessage": {
      "Action": "OPEN_APP",
      "Body": "Hello from AWS End User Messaging Push! This is an APNS
 notification.",
      "Priority": "normal",
      "SilentPush": false,
      "Title": "My First APNS Push Notification",
      "TimeToLive": 30,
      "Badge": 1,
      "Sound": "default"
    }
  }
}
EOF

echo "Message files created:"
echo "- FCM message file (for FCM/Android)"
echo "- APNS message file (for APNS/iOS)"
echo ""
echo "Note: These messages use placeholder device tokens and will not actually be
 delivered."
```

```
echo "To send real messages, you would need to replace the sample device tokens
 with actual ones."

# Step 5: Demonstrate how to send messages (this will fail with placeholder
 tokens)
echo ""
echo "========================================="
echo "DEMONSTRATING MESSAGE SENDING"
echo "========================================="
echo "Attempting to send FCM message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<gcm-message>"
SEND_FCM_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$GCM_MESSAGE_FILE" 2>&1)

if check_error "$SEND_FCM_OUTPUT" "send-messages (FCM)" "true"; then
    echo "FCM message sent successfully:"
    echo "$SEND_FCM_OUTPUT"
else
    echo "As expected, FCM message sending failed with placeholder token."
    echo "Error details: $SEND_FCM_OUTPUT"
fi

echo ""
echo "Attempting to send APNS message (will fail with placeholder token)..."

echo "Executing: aws pinpoint send-messages --application-id $APP_ID --message-
request file://<apns-message>"
SEND_APNS_OUTPUT=$(aws pinpoint send-messages \
    --application-id "$APP_ID" \
    --message-request "file://$APNS_MESSAGE_FILE" 2>&1)

if check_error "$SEND_APNS_OUTPUT" "send-messages (APNS)" "true"; then
    echo "APNS message sent successfully:"
    echo "$SEND_APNS_OUTPUT"
else
    echo "As expected, APNS message sending failed with placeholder token."
    echo "Error details: $SEND_APNS_OUTPUT"
fi

# Step 6: Show application details
echo ""
```

```
echo "============================================"
echo "APPLICATION DETAILS"
echo "============================================"
echo "Retrieving application details..."

echo "Executing: aws pinpoint get-app --application-id $APP_ID"
GET_APP_OUTPUT=$(aws pinpoint get-app --application-id "$APP_ID" 2>&1)
if check_error "$GET_APP_OUTPUT" "get-app"; then
    echo "Application details:"
    echo "$GET_APP_OUTPUT"
fi

# Display summary of created resources
echo ""
echo "============================================"
echo "RESOURCES CREATED"
echo "============================================"
echo "AWS Resources:"
for resource in "${AWS_RESOURCES[@]+"${AWS_RESOURCES[@]}"}"; do
    echo "- Application: $resource"
done

echo ""
echo "Files created:"
echo "- $LOG_FILE (script log)"

# Auto-cleanup information
echo ""
echo "============================================"
echo "CLEANUP INFORMATION"
echo "============================================"
echo "This script created AWS resources that may incur charges."
echo "AWS resources will be automatically cleaned up on script exit."
echo ""
echo "To manually delete resources later, use:"
echo "  aws pinpoint delete-app --application-id $APP_ID"

# Set flag to delete AWS resources on cleanup
DELETE_AWS_RESOURCES=true

echo ""
echo "============================================"
echo "SCRIPT COMPLETED SUCCESSFULLY"
echo "============================================"
```

```
echo "This script demonstrated:"
echo "1. Creating an AWS End User Messaging Push application"
echo "2. Attempting to enable FCM and APNS channels (with placeholder
 credentials)"
echo "3. Creating message templates for different platforms"
echo "4. Demonstrating message sending commands (with placeholder tokens)"
echo "5. Retrieving application details"
echo "6. Proper cleanup of resources"
echo ""
echo "Security best practices implemented:"
echo "- Secure temporary file handling with restricted permissions"
echo "- Input validation for application names"
echo "- Error handling with proper exit codes"
echo "- Credential validation before AWS operations"
echo "- Automatic cleanup on script exit"
echo "- Secure file destruction for sensitive data"
echo ""
echo "For production use:"
echo "- Replace placeholder API keys with real FCM server keys"
echo "- Replace placeholder certificates with real APNS certificates"
echo "- Replace placeholder device tokens with real device tokens"
echo "- Use AWS IAM roles instead of long-term credentials"
echo "- Implement comprehensive error handling"
echo "- Store sensitive credentials in AWS Secrets Manager or Parameter Store"
echo "- Enable CloudTrail for audit logging"
echo "- Use VPC endpoints for private AWS API access"
echo ""
echo "Log file: $LOG_FILE"
echo "Script completed at: $(date)"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - CreateApp
  - DeleteApp
  - GetApp
  - GetApps
  - GetCallerIdentity
  - SendMessages
  - UpdateApnsChannel
  - UpdateGcmChannel

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

IoT Core 入门

以下代码示例展示了如何：

- 创建 IoT 资源
- 配置您的设备
- 运行示例应用程序
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS IoT Core Getting Started Script
# This script creates AWS IoT resources, configures a device, and runs a sample
 application

# Set up logging
LOG_FILE="iot-core-setup.log"
echo "Starting AWS IoT Core setup at $(date)" > $LOG_FILE

# Function to log commands and their outputs
log_cmd() {
    echo "$(date): Running command: $1" >> $LOG_FILE
    eval "$1" 2>&1 | tee -a $LOG_FILE
    return ${PIPESTATUS[0]}
}

# Function to check for errors
```

```
check_error() {
    if [ $1 -ne 0 ]; then
        echo "ERROR: Command failed with exit code $1" | tee -a $LOG_FILE
        echo "Please check the log file $LOG_FILE for details" | tee -a $LOG_FILE
        cleanup_on_error
        exit $1
    fi
}

# Function to cleanup resources on error
cleanup_on_error() {
    echo "Error encountered. Attempting to clean up resources..." | tee -a
 $LOG_FILE
    echo "Resources created:" | tee -a $LOG_FILE
    if [ ! -z "$CERTIFICATE_ARN" ]; then
        echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
        if [ ! -z "$POLICY_NAME" ]; then
            log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
        fi
        if [ ! -z "$THING_NAME" ]; then
            log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --
principal $CERTIFICATE_ARN"
        fi
        if [ ! -z "$CERTIFICATE_ID" ]; then
            log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID
 --new-status INACTIVE"
            log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"
        fi
    fi
    if [ ! -z "$THING_NAME" ]; then
        echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-thing --thing-name $THING_NAME"
    fi
    if [ ! -z "$POLICY_NAME" ]; then
        echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    fi
    if [ ! -z "$SHARED_POLICY_NAME" ]; then
        echo "Shared Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
        log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"
    fi
}
```

```
# Generate unique identifiers
RANDOM_SUFFIX=$(openssl rand -hex 4)
THING_NAME="MyIoTThing-${RANDOM_SUFFIX}"
POLICY_NAME="MyIoTPolicy-${RANDOM_SUFFIX}"
SHARED_POLICY_NAME="SharedSubPolicy-${RANDOM_SUFFIX}"
CERTS_DIR="$HOME/certs"

echo "================================================" | tee -a $LOG_FILE
echo "AWS IoT Core Getting Started" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "This script will:" | tee -a $LOG_FILE
echo "1. Create AWS IoT resources (policy, thing, certificate)" | tee -a
 $LOG_FILE
echo "2. Configure your device" | tee -a $LOG_FILE
echo "3. Set up for running the sample application" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Certificates Directory: $CERTS_DIR" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE

# Get AWS account ID
echo "Getting AWS account ID..." | tee -a $LOG_FILE
ACCOUNT_ID=$(log_cmd "aws sts get-caller-identity --query Account --output text")
check_error $?

# Get AWS region
echo "Getting AWS region..." | tee -a $LOG_FILE
REGION=$(log_cmd "aws configure get region")
check_error $?
if [ -z "$REGION" ]; then
    echo "AWS region not configured. Please run 'aws configure' to set your
 region." | tee -a $LOG_FILE
    exit 1
fi

echo "Using AWS Account ID: $ACCOUNT_ID and Region: $REGION" | tee -a $LOG_FILE

# Step 1: Create AWS IoT Resources
echo "" | tee -a $LOG_FILE
echo "Step 1: Creating AWS IoT Resources..." | tee -a $LOG_FILE

# Create IoT policy
```

```
echo "Creating IoT policy document..." | tee -a $LOG_FILE
cat > iot-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/test-*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic"
      ]
    }
  ]
}
EOF

echo "Creating IoT policy: $POLICY_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-policy --policy-name $POLICY_NAME --policy-
document file://iot-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

# Create IoT thing
```

```
echo "Creating IoT thing: $THING_NAME..." | tee -a $LOG_FILE
log_cmd "aws iot create-thing --thing-name $THING_NAME"
check_error $?

THING_ARN=$(aws iot describe-thing --thing-name "$THING_NAME" --query 'thingArn'
 --output text)
aws iot tag-resource --resource-arn "$THING_ARN" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-iot-core-gs

# Create directory for certificates
echo "Creating certificates directory..." | tee -a $LOG_FILE
log_cmd "mkdir -p $CERTS_DIR"
check_error $?

# Create keys and certificate
echo "Creating keys and certificate..." | tee -a $LOG_FILE
CERT_OUTPUT=$(log_cmd "aws iot create-keys-and-certificate --set-as-active
 --certificate-pem-outfile $CERTS_DIR/device.pem.crt --public-key-outfile
 $CERTS_DIR/public.pem.key --private-key-outfile $CERTS_DIR/private.pem.key")
check_error $?

# Extract certificate ARN and ID
CERTIFICATE_ARN=$(echo "$CERT_OUTPUT" | grep "certificateArn" | cut -d'"' -f4)
CERTIFICATE_ID=$(echo "$CERTIFICATE_ARN" | cut -d/ -f2)

if [ -z "$CERTIFICATE_ARN" ] || [ -z "$CERTIFICATE_ID" ]; then
    echo "Failed to extract certificate ARN or ID" | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE

# Attach policy to certificate
echo "Attaching policy to certificate..." | tee -a $LOG_FILE
log_cmd "aws iot attach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Attach certificate to thing
echo "Attaching certificate to thing..." | tee -a $LOG_FILE
log_cmd "aws iot attach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"
```

```
check_error $?

# Download Amazon Root CA certificate
echo "Downloading Amazon Root CA certificate..." | tee -a $LOG_FILE
log_cmd "curl -s -o $CERTS_DIR/Amazon-root-CA-1.pem https://www.amazontrust.com/
repository/AmazonRootCA1.pem"
check_error $?

# Step 2: Configure Your Device
echo "" | tee -a $LOG_FILE
echo "Step 2: Configuring Your Device..." | tee -a $LOG_FILE

# Check if Git is installed
echo "Checking if Git is installed..." | tee -a $LOG_FILE
if ! command -v git &> /dev/null; then
    echo "Git is not installed. Please install Git and run this script again." |
 tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Check if Python is installed
echo "Checking if Python is installed..." | tee -a $LOG_FILE
if ! command -v python3 &> /dev/null; then
    echo "Python 3 is not installed. Please install Python 3 and run this script
 again." | tee -a $LOG_FILE
    cleanup_on_error
    exit 1
fi

# Install AWS IoT Device SDK for Python
echo "Installing AWS IoT Device SDK for Python..." | tee -a $LOG_FILE
log_cmd "python3 -m pip install awsiotsdk"
check_error $?

# Clone the AWS IoT Device SDK for Python repository
echo "Cloning AWS IoT Device SDK for Python repository..." | tee -a $LOG_FILE
if [ ! -d "$HOME/aws-iot-device-sdk-python-v2" ]; then
    log_cmd "cd $HOME && git clone https://github.com/aws/aws-iot-device-sdk-
python-v2.git"
    check_error $?
else
    echo "AWS IoT Device SDK for Python repository already exists." | tee -a
 $LOG_FILE
```

```
fi

# Step 3: Get AWS IoT Endpoint
echo "" | tee -a $LOG_FILE
echo "Step 3: Getting AWS IoT Endpoint..." | tee -a $LOG_FILE

IOT_ENDPOINT=$(log_cmd "aws iot describe-endpoint --endpoint-type iot:Data-ATS --
query endpointAddress --output text")
check_error $?

echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE

# Create a shared subscription policy (optional)
echo "" | tee -a $LOG_FILE
echo "Creating shared subscription policy (optional)..." | tee -a $LOG_FILE

cat > shared-sub-policy.json << EOF
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iot:Connect"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:client/*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Publish",
        "iot:Receive"
      ],
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topic/test/topic"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "iot:Subscribe"
      ],
```

```
      "Resource": [
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/test/topic",
        "arn:aws:iot:$REGION:$ACCOUNT_ID:topicfilter/\$share/*/test/topic"
      ]
    }
  ]
}
EOF

log_cmd "aws iot create-policy --policy-name $SHARED_POLICY_NAME --policy-
document file://shared-sub-policy.json --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-iot-core-gs"
check_error $?

log_cmd "aws iot attach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"
check_error $?

# Summary of created resources
echo "" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "Setup Complete! Resources Created:" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "Thing Name: $THING_NAME" | tee -a $LOG_FILE
echo "Policy Name: $POLICY_NAME" | tee -a $LOG_FILE
echo "Shared Subscription Policy Name: $SHARED_POLICY_NAME" | tee -a $LOG_FILE
echo "Certificate ID: $CERTIFICATE_ID" | tee -a $LOG_FILE
echo "Certificate ARN: $CERTIFICATE_ARN" | tee -a $LOG_FILE
echo "Certificate Files Location: $CERTS_DIR" | tee -a $LOG_FILE
echo "AWS IoT Endpoint: $IOT_ENDPOINT" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE

# Instructions for running the sample application
echo "" | tee -a $LOG_FILE
echo "To run the sample application, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
echo "python3 pubsub.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key" | tee -a $LOG_FILE
echo "" | tee -a $LOG_FILE
echo "To run the shared subscription example, execute:" | tee -a $LOG_FILE
echo "cd $HOME/aws-iot-device-sdk-python-v2/samples" | tee -a $LOG_FILE
```

```
echo "python3 mqtt5_shared_subscription.py \\" | tee -a $LOG_FILE
echo "  --endpoint $IOT_ENDPOINT \\" | tee -a $LOG_FILE
echo "  --ca_file $CERTS_DIR/Amazon-root-CA-1.pem \\" | tee -a $LOG_FILE
echo "  --cert $CERTS_DIR/device.pem.crt \\" | tee -a $LOG_FILE
echo "  --key $CERTS_DIR/private.pem.key \\" | tee -a $LOG_FILE
echo "  --group_identifier consumer" | tee -a $LOG_FILE

# Ask if user wants to clean up resources
echo "" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "CLEANUP CONFIRMATION" | tee -a $LOG_FILE
echo "================================================" | tee -a $LOG_FILE
echo "Do you want to clean up all created resources? (y/n): " | tee -a $LOG_FILE
read -r CLEANUP_CHOICE

if [[ $CLEANUP_CHOICE =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a $LOG_FILE

    # Detach policies from certificate
    echo "Detaching policies from certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN"
    log_cmd "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN"

    # Detach certificate from thing
    echo "Detaching certificate from thing..." | tee -a $LOG_FILE
    log_cmd "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN"

    # Update certificate status to INACTIVE
    echo "Setting certificate to inactive..." | tee -a $LOG_FILE
    log_cmd "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE"

    # Delete certificate
    echo "Deleting certificate..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-certificate --certificate-id $CERTIFICATE_ID"

    # Delete thing
    echo "Deleting thing..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-thing --thing-name $THING_NAME"

    # Delete policies
```

```
    echo "Deleting policies..." | tee -a $LOG_FILE
    log_cmd "aws iot delete-policy --policy-name $POLICY_NAME"
    log_cmd "aws iot delete-policy --policy-name $SHARED_POLICY_NAME"

    echo "Cleanup complete!" | tee -a $LOG_FILE
else
    echo "Resources were not cleaned up. You can manually clean them up later." |
 tee -a $LOG_FILE
    echo "To clean up resources, run the following commands:" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-policy --policy-name $SHARED_POLICY_NAME --target
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot detach-thing-principal --thing-name $THING_NAME --principal
 $CERTIFICATE_ARN" | tee -a $LOG_FILE
    echo "aws iot update-certificate --certificate-id $CERTIFICATE_ID --new-
status INACTIVE" | tee -a $LOG_FILE
    echo "aws iot delete-certificate --certificate-id $CERTIFICATE_ID" | tee -a
 $LOG_FILE
    echo "aws iot delete-thing --thing-name $THING_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $POLICY_NAME" | tee -a $LOG_FILE
    echo "aws iot delete-policy --policy-name $SHARED_POLICY_NAME" | tee -a
 $LOG_FILE
fi

echo "" | tee -a $LOG_FILE
echo "Script execution completed. See $LOG_FILE for details." | tee -a $LOG_FILE
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachPolicy
  - AttachThingPrincipal
  - CreateKeysAndCertificate
  - CreatePolicy
  - CreateThing
  - DeleteCertificate
  - DeletePolicy
  - DeleteThing
  - DescribeEndpoint

- DetachPolicy
- DetachThingPrincipal
- GetCallerIdentity
- UpdateCertificate

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

WAF 入门

以下代码示例展示了如何：

- 创建 Web ACL
- 添加字符串匹配规则
- 添加托管规则。
- 配置 日志记录
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS WAF Getting Started Script
# This script creates a Web ACL with a string match rule and AWS Managed Rules,
# associates it with a CloudFront distribution, and then cleans up all resources.

set -euo pipefail

# Security: Restrict file permissions
```

```
umask 077

# Set up logging with secure file handling
LOG_FILE="waf-tutorial.log"
touch "$LOG_FILE"
chmod 600 "$LOG_FILE"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "================================================="
echo "AWS WAF Getting Started Tutorial"
echo "================================================="
echo "This script will create AWS WAF resources and associate"
echo "them with a CloudFront distribution."
echo ""

# Maximum number of retries for operations
MAX_RETRIES=3

# Function to handle errors securely
handle_error() {
    local error_msg="$1"
    echo "ERROR: $error_msg" >&2
    echo "Check the log file for details: $LOG_FILE" >&2
    cleanup_resources
    exit 1
}

# Function to validate AWS CLI JSON output
validate_json() {
    local json_string="$1"
    if ! echo "$json_string" | jq empty 2>/dev/null; then
        return 1
    fi
    return 0
}

# Function to safely extract JSON values
extract_json_value() {
    local json_string="$1"
    local key_path="$2"

    if ! validate_json "$json_string"; then
        return 1
    fi
```

```
    echo "$json_string" | jq -r "$key_path" 2>/dev/null || return 1
}

# Function to clean up resources securely
cleanup_resources() {
    echo ""
    echo "=================================================="
    echo "CLEANING UP RESOURCES"
    echo "=================================================="

    if [ -n "${DISTRIBUTION_ID:-}" ] && [ -n "${WEB_ACL_ARN:-}" ]; then
        echo "Disassociating Web ACL from CloudFront distribution..."
        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID"
            return
        fi

        local disassociate_result
        disassociate_result=$(aws wafv2 disassociate-web-acl \
            --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
            --region us-east-1 2>&1) || true

        if echo "$disassociate_result" | grep -qi "error"; then
            echo "Warning: Failed to disassociate Web ACL: $disassociate_result"
        else
            echo "Web ACL disassociated successfully."
        fi
    fi

    if [ -n "${WEB_ACL_ID:-}" ] && [ -n "${WEB_ACL_NAME:-}" ]; then
        echo "Deleting Web ACL..."

        local get_result
        get_result=$(aws wafv2 get-web-acl \
            --name "$WEB_ACL_NAME" \
            --scope CLOUDFRONT \
            --id "$WEB_ACL_ID" \
            --region us-east-1 2>&1) || true
```

```
        if echo "$get_result" | grep -qi "error"; then
            echo "Warning: Failed to get Web ACL for deletion: $get_result"
            echo "You may need to manually delete the Web ACL using the AWS
 Console."
        else
            local latest_token
            latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken'
 2>/dev/null) || latest_token=""

            if [ -n "$latest_token" ]; then
                local delete_result
                delete_result=$(aws wafv2 delete-web-acl \
                    --name "$WEB_ACL_NAME" \
                    --scope CLOUDFRONT \
                    --id "$WEB_ACL_ID" \
                    --lock-token "$latest_token" \
                    --region us-east-1 2>&1) || true

                if echo "$delete_result" | grep -qi "error"; then
                    echo "Warning: Failed to delete Web ACL: $delete_result"
                    echo "You may need to manually delete the Web ACL using the
 AWS Console."
                else
                    echo "Web ACL deleted successfully."
                fi
            else
                echo "Warning: Could not extract lock token for deletion. You may
 need to manually delete the Web ACL."
            fi
        fi
    fi

    echo "Cleanup process completed."
}

# Security: Trap EXIT to ensure cleanup on any exit
trap cleanup_resources EXIT

# Generate a random identifier for resource names using secure method
RANDOM_ID=$(openssl rand -hex 4) || handle_error "Failed to generate random ID"
WEB_ACL_NAME="MyWebACL-${RANDOM_ID}"
METRIC_NAME="MyWebACLMetrics-${RANDOM_ID}"
```

```
echo "Using Web ACL name: $WEB_ACL_NAME"

# Step 1: Create a Web ACL
echo ""
echo "================================================="
echo "STEP 1: Creating Web ACL"
echo "================================================="

local create_result
create_result=$(aws wafv2 create-web-acl \
    --name "$WEB_ACL_NAME" \
    --scope "CLOUDFRONT" \
    --default-action Allow={} \
    --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-waf-gs \
    --region us-east-1 2>&1) || handle_error "Failed to create Web ACL"

if ! validate_json "$create_result"; then
    handle_error "Invalid JSON response from create-web-acl"
fi

# Extract Web ACL ID, ARN, and Lock Token from the response
WEB_ACL_ID=$(extract_json_value "$create_result" '.Summary.Id') || handle_error
 "Failed to extract Web ACL ID"
WEB_ACL_ARN=$(extract_json_value "$create_result" '.Summary.ARN') || handle_error
 "Failed to extract Web ACL ARN"
LOCK_TOKEN=$(extract_json_value "$create_result" '.Summary.LockToken') ||
 handle_error "Failed to extract Lock Token"

echo "Web ACL created successfully with ID: $WEB_ACL_ID"
echo "Lock Token: [REDACTED]"

# Step 2: Add a String Match Rule
echo ""
echo "================================================="
echo "STEP 2: Adding String Match Rule"
echo "================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add string match rule..."

    # Get the latest lock token before updating
```

```
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
```

```
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add string match rule after $MAX_RETRIES
 attempts"
        fi
        sleep 2
```

```
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add string match rule: $update_result"
    else
        echo "String match rule added successfully."
        break
    fi
done

# Step 3: Add AWS Managed Rules
echo ""
echo "=================================================="
echo "STEP 3: Adding AWS Managed Rules"
echo "=================================================="

for ((i=1; i<=MAX_RETRIES; i++)); do
    echo "Attempt $i to add AWS Managed Rules..."

    # Get the latest lock token before updating
    local get_result
    get_result=$(aws wafv2 get-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope CLOUDFRONT \
        --id "$WEB_ACL_ID" \
        --region us-east-1 2>&1) || true

    if echo "$get_result" | grep -qi "error"; then
        echo "Warning: Failed to get Web ACL for update: $get_result"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to get Web ACL after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi

    if ! validate_json "$get_result"; then
        echo "Warning: Invalid JSON response from get-web-acl"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Invalid JSON response after $MAX_RETRIES attempts"
        fi
        sleep 2
        continue
    fi
```

```
    local latest_token
    latest_token=$(extract_json_value "$get_result" '.WebACL.LockToken' 2>/dev/
null) || true

    if [ -z "$latest_token" ]; then
        echo "Warning: Could not extract lock token for update"
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to extract lock token after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    fi

    local update_result
    update_result=$(aws wafv2 update-web-acl \
        --name "$WEB_ACL_NAME" \
        --scope "CLOUDFRONT" \
        --id "$WEB_ACL_ID" \
        --lock-token "$latest_token" \
        --default-action Allow={} \
        --rules '[{
            "Name": "UserAgentRule",
            "Priority": 0,
            "Statement": {
                "ByteMatchStatement": {
                    "SearchString": "MyAgent",
                    "FieldToMatch": {
                        "SingleHeader": {
                            "Name": "user-agent"
                        }
                    },
                    "TextTransformations": [
                        {
                            "Priority": 0,
                            "Type": "NONE"
                        }
                    ],
                    "PositionalConstraint": "EXACTLY"
                }
            },
            "Action": {
                "Count": {}
            },
```

```
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "UserAgentRuleMetric"
            }
        },
        {
            "Name": "AWS-AWSManagedRulesCommonRuleSet",
            "Priority": 1,
            "Statement": {
                "ManagedRuleGroupStatement": {
                    "VendorName": "AWS",
                    "Name": "AWSManagedRulesCommonRuleSet",
                    "ExcludedRules": []
                }
            },
            "OverrideAction": {
                "Count": {}
            },
            "VisibilityConfig": {
                "SampledRequestsEnabled": true,
                "CloudWatchMetricsEnabled": true,
                "MetricName": "AWS-AWSManagedRulesCommonRuleSet"
            }
        }]' \
        --visibility-config
 "SampledRequestsEnabled=true,CloudWatchMetricsEnabled=true,MetricName=
$METRIC_NAME" \
        --region us-east-1 2>&1) || true

    if echo "$update_result" | grep -qi "WAFOptimisticLockException"; then
        echo "Optimistic lock exception encountered. Will retry with new lock
 token."
        if [ "$i" -eq "$MAX_RETRIES" ]; then
            handle_error "Failed to add AWS Managed Rules after $MAX_RETRIES
 attempts"
        fi
        sleep 2
        continue
    elif echo "$update_result" | grep -qi "error"; then
        handle_error "Failed to add AWS Managed Rules: $update_result"
    else
        echo "AWS Managed Rules added successfully."
        break
```

```
    fi
done

# Step 4: List CloudFront distributions
echo ""
echo "=================================================="
echo "STEP 4: Listing CloudFront Distributions"
echo "=================================================="

local cf_result
cf_result=$(aws cloudfront list-distributions --query "DistributionList.Items[*].
{Id:Id,DomainName:DomainName}" --output table 2>&1) || cf_result=""

if echo "$cf_result" | grep -qi "error"; then
    echo "Warning: Failed to list CloudFront distributions: $cf_result"
    echo "Continuing without CloudFront association."
    DISTRIBUTION_ID=""
else
    echo "$cf_result"

    # Auto-select first CloudFront distribution if available
    echo ""
    echo "=================================================="
    echo "STEP 5: Associate Web ACL with CloudFront Distribution"
    echo "=================================================="

    local first_dist
    first_dist=$(aws cloudfront list-distributions --query
 "DistributionList.Items[0].Id" --output text 2>&1) || first_dist=""

    if [ -n "$first_dist" ] && [ "$first_dist" != "None" ] && ! echo
 "$first_dist" | grep -qi "error"; then
        DISTRIBUTION_ID="$first_dist"
        echo "Using CloudFront distribution: $DISTRIBUTION_ID"

        local account_id
        account_id=$(aws sts get-caller-identity --query Account --output text
 2>/dev/null) || account_id=""

        if [ -z "$account_id" ]; then
            echo "Warning: Could not retrieve AWS account ID for association"
            DISTRIBUTION_ID=""
        else
            local associate_result
```

```
            associate_result=$(aws wafv2 associate-web-acl \
                --web-acl-arn "$WEB_ACL_ARN" \
                --resource-arn "arn:aws:cloudfront::${account_id}:distribution/
${DISTRIBUTION_ID}" \
                --region us-east-1 2>&1) || true

            if echo "$associate_result" | grep -qi "error"; then
                echo "Warning: Failed to associate Web ACL with CloudFront
 distribution: $associate_result"
                echo "Continuing without CloudFront association."
                DISTRIBUTION_ID=""
            else
                echo "Web ACL associated with CloudFront distribution
 successfully."
            fi
        fi
    else
        echo "No CloudFront distributions available. Skipping association."
        DISTRIBUTION_ID=""
    fi
fi

# Display summary of created resources
echo ""
echo "================================================="
echo "RESOURCE SUMMARY"
echo "================================================="
echo "Web ACL Name: $WEB_ACL_NAME"
echo "Web ACL ID: $WEB_ACL_ID"
echo "Web ACL ARN: $WEB_ACL_ARN"
if [ -n "${DISTRIBUTION_ID:-}" ]; then
    echo "Associated CloudFront Distribution: $DISTRIBUTION_ID"
fi
echo ""

# Auto-confirm cleanup
echo "================================================="
echo "CLEANUP CONFIRMATION"
echo "================================================="
echo "Proceeding with automatic cleanup of all created resources..."

echo ""
echo "================================================="
echo "Tutorial completed!"
```

```
echo "============================================================="
echo "Log file: $LOG_FILE"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AssociateWebAcl
  - CreateWebAcl
  - DeleteWebAcl
  - DisassociateWebAcl
  - GetCallerIdentity
  - GetWebAcl
  - ListDistributions
  - UpdateWebAcl

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将硬编码的密钥移动到 Secrets Manager

以下代码示例展示了如何：

- 创建 IAM 角色
- 在 Secrets Manager 中创建密钥
- 更新您的应用程序代码。
- 更新密钥
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# Script to move hardcoded secrets to AWS Secrets Manager
# This script demonstrates how to create IAM roles, store a secret in AWS Secrets
 Manager,
# and set up appropriate permissions

set -euo pipefail

# Set up logging
LOG_FILE="secrets_manager_tutorial.log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS Secrets Manager tutorial script at $(date)"
echo "======================================================="

# Function to check for errors in command output
check_error() {
    local output=$1
    local cmd=$2

    if echo "$output" | grep -qi "error"; then
        echo "ERROR: Command failed: $cmd"
        echo "$output"
        cleanup_resources
        exit 1
    fi
}

# Function to generate a random identifier using secure method
generate_random_id() {
    python3 -c "import secrets; print('sm' + secrets.token_hex(4))"
}

# Function to safely clean up resources
cleanup_resources() {
    echo ""
    echo "==========================================="
    echo "RESOURCES CREATED"
    echo "==========================================="

    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Secret: $SECRET_NAME"
```

```
    fi

    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "IAM Role: $RUNTIME_ROLE_NAME"
    fi

    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "IAM Role: $ADMIN_ROLE_NAME"
    fi

    echo ""
    echo "==========================================="
    echo "CLEANUP CONFIRMATION"
    echo "==========================================="
    echo "Cleaning up all created resources..."

    # Delete secret if it exists
    if [ -n "${SECRET_NAME:-}" ]; then
        echo "Deleting secret: $SECRET_NAME"
        aws secretsmanager delete-secret --secret-id "$SECRET_NAME" --force-
delete-without-recovery 2>/dev/null || true
    fi

    # Detach policies and delete runtime role if it exists
    if [ -n "${RUNTIME_ROLE_NAME:-}" ]; then
        echo "Deleting inline policies from runtime role: $RUNTIME_ROLE_NAME"
        for policy in $(aws iam list-role-policies --role-name
 "$RUNTIME_ROLE_NAME" --query 'PolicyNames[]' --output text 2>/dev/null || true);
 do
            aws iam delete-role-policy --role-name "$RUNTIME_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done
        echo "Deleting IAM role: $RUNTIME_ROLE_NAME"
        aws iam delete-role --role-name "$RUNTIME_ROLE_NAME" 2>/dev/null || true
    fi

    # Detach policies and delete admin role if it exists
    if [ -n "${ADMIN_ROLE_NAME:-}" ]; then
        echo "Detaching policy from role: $ADMIN_ROLE_NAME"
        aws iam detach-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>/dev/null || true

        for policy in $(aws iam list-role-policies --role-name "$ADMIN_ROLE_NAME"
 --query 'PolicyNames[]' --output text 2>/dev/null || true); do
```

```
            aws iam delete-role-policy --role-name "$ADMIN_ROLE_NAME" --policy-
name "$policy" 2>/dev/null || true
        done

        echo "Deleting IAM role: $ADMIN_ROLE_NAME"
        aws iam delete-role --role-name "$ADMIN_ROLE_NAME" 2>/dev/null || true
    fi

    echo "Cleanup completed."
}

# Trap to ensure cleanup on script exit
trap 'echo "Script interrupted. Running cleanup..."; cleanup_resources' INT TERM
 EXIT

# Generate random identifiers for resources
ADMIN_ROLE_NAME="SecretsManagerAdmin-$(generate_random_id)"
RUNTIME_ROLE_NAME="RoleToRetrieveSecretAtRuntime-$(generate_random_id)"
SECRET_NAME="MyAPIKey-$(generate_random_id)"

echo "Using the following resource names:"
echo "Admin Role: $ADMIN_ROLE_NAME"
echo "Runtime Role: $RUNTIME_ROLE_NAME"
echo "Secret Name: $SECRET_NAME"
echo ""

# Step 1: Create IAM roles
echo "Creating IAM roles..."

# Create assume role policy document
ASSUME_ROLE_POLICY=$(cat <<'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "ec2.amazonaws.com"
            },
            "Action": "sts:AssumeRole"
        }
    ]
}
EOF
```

```
)

# Create the SecretsManagerAdmin role
echo "Creating admin role: $ADMIN_ROLE_NAME"
ADMIN_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$ADMIN_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$ADMIN_ROLE_OUTPUT" "create-role for admin"
echo "$ADMIN_ROLE_OUTPUT"
aws iam tag-role --role-name "$ADMIN_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Attach the SecretsManagerReadWrite policy to the admin role
echo "Attaching SecretsManagerReadWrite policy to admin role"
ATTACH_POLICY_OUTPUT=$(aws iam attach-role-policy \
    --role-name "$ADMIN_ROLE_NAME" \
    --policy-arn "arn:aws:iam::aws:policy/SecretsManagerReadWrite" 2>&1)

check_error "$ATTACH_POLICY_OUTPUT" "attach-role-policy for admin"
echo "Policy attached successfully"

# Create the RoleToRetrieveSecretAtRuntime role
echo "Creating runtime role: $RUNTIME_ROLE_NAME"
RUNTIME_ROLE_OUTPUT=$(aws iam create-role \
    --role-name "$RUNTIME_ROLE_NAME" \
    --assume-role-policy-document "$ASSUME_ROLE_POLICY" 2>&1)

check_error "$RUNTIME_ROLE_OUTPUT" "create-role for runtime"
echo "$RUNTIME_ROLE_OUTPUT"
aws iam tag-role --role-name "$RUNTIME_ROLE_NAME" --tags Key=project,Value=doc-
smith Key=tutorial,Value=aws-secrets-manager-gs

# Wait for roles to be fully created
echo "Waiting for IAM roles to be fully created..."
sleep 10

# Step 2: Create a secret in AWS Secrets Manager
echo "Creating secret in AWS Secrets Manager..."

# Generate secure secret value using environment variable or secure method
# WARNING: In production, use secure methods to inject secrets (AWS CodeBuild,
 parameter store, etc.)
if [ -z "${TUTORIAL_SECRET_VALUE:-}" ]; then
```

```
    SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_client_id','ClientSecret':__import__('secrets').token_u
else
    SECRET_VALUE="$TUTORIAL_SECRET_VALUE"
fi

CREATE_SECRET_OUTPUT=$(aws secretsmanager create-secret \
    --name "$SECRET_NAME" \
    --description "API key for my application" \
    --secret-string "$SECRET_VALUE" \
    --add-replica-regions 'Region=us-east-1' \
    --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-secrets-manager-gs
 2>&1)

check_error "$CREATE_SECRET_OUTPUT" "create-secret"
echo "$CREATE_SECRET_OUTPUT"

# Get AWS account ID
echo "Getting AWS account ID..."
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text 2>&1)
check_error "$ACCOUNT_ID" "get-caller-identity"
echo "Account ID: $ACCOUNT_ID"

# Get secret ARN for precise resource policy
echo "Getting secret ARN..."
SECRET_ARN=$(aws secretsmanager describe-secret \
    --secret-id "$SECRET_NAME" \
    --query 'ARN' \
    --output text 2>&1)
check_error "$SECRET_ARN" "describe-secret"

# Add resource policy to the secret with least privilege
echo "Adding resource policy to secret..."
RESOURCE_POLICY=$(cat <<EOF
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowRuntimeRoleReadSecret",
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::$ACCOUNT_ID:role/$RUNTIME_ROLE_NAME"
            },
            "Action": "secretsmanager:GetSecretValue",
```

```
            "Resource": "$SECRET_ARN",
            "Condition": {
                "StringEquals": {
                    "secretsmanager:VersionStage": "AWSCURRENT"
                }
            }
        }
    ]
}
EOF
)

PUT_POLICY_OUTPUT=$(aws secretsmanager put-resource-policy \
    --secret-id "$SECRET_NAME" \
    --resource-policy "$RESOURCE_POLICY" \
    --block-public-policy 2>&1)

check_error "$PUT_POLICY_OUTPUT" "put-resource-policy"
echo "Resource policy added successfully"

# Enable rotation policy recommendation
echo "Enabling secret metadata tags for rotation tracking..."
aws secretsmanager tag-resource \
    --secret-id "$SECRET_NAME" \
    --tags Key=Purpose,Value=Tutorial Key=AutoRotation,Value=Recommended 2>/dev/
null || true

# Step 3: Demonstrate retrieving the secret
echo "Retrieving the secret value (for demonstration purposes)..."
GET_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$GET_SECRET_OUTPUT" "get-secret-value"
echo "Secret retrieved successfully. Secret metadata:"
echo "$GET_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 4: Update the secret with new values
echo "Updating the secret with new values..."
UPDATE_SECRET_VALUE=$(python3 -c "import json;
 print(json.dumps({'ClientID':'my_new_client_id','ClientSecret':__import__('secrets').tok

UPDATE_SECRET_OUTPUT=$(aws secretsmanager update-secret \
```

```
    --secret-id "$SECRET_NAME" \
    --secret-string "$UPDATE_SECRET_VALUE" 2>&1)

check_error "$UPDATE_SECRET_OUTPUT" "update-secret"
echo "Secret updated successfully"

# Step 5: Verify the updated secret
echo "Verifying the updated secret..."
VERIFY_SECRET_OUTPUT=$(aws secretsmanager get-secret-value \
    --secret-id "$SECRET_NAME" 2>&1)

check_error "$VERIFY_SECRET_OUTPUT" "get-secret-value for verification"
echo "Updated secret retrieved successfully. Secret metadata:"
echo "$VERIFY_SECRET_OUTPUT" | jq '{ARN: .ARN, Name: .Name,
 LastUpdatedDate: .LastUpdatedDate, VersionIdsToStages: .VersionIdsToStages}' 2>/
dev/null || echo "Secret metadata retrieved (jq not available)"

# Step 6: Display rotation recommendations
echo ""
echo "Rotation Configuration Recommendations:"
echo "======================================="
DESCRIBE_OUTPUT=$(aws secretsmanager describe-secret --secret-id "$SECRET_NAME"
 2>&1)
if echo "$DESCRIBE_OUTPUT" | grep -q "RotationRules"; then
    echo "Current rotation configuration:"
    echo "$DESCRIBE_OUTPUT" | jq '.RotationRules' 2>/dev/null || echo "Rotation
 rules available"
else
    echo "No automatic rotation configured. Consider enabling rotation with:"
    echo "aws secretsmanager rotate-secret --secret-id $SECRET_NAME --rotation-
lambda-arn arn:aws:lambda:REGION:ACCOUNT:function:FUNCTION_NAME --rotation-rules
 AutomaticallyAfterDays=30"
fi

echo ""
echo "======================================================"
echo "Tutorial completed successfully!"
echo ""
echo "Summary of what we did:"
echo "1. Created IAM roles for managing and retrieving secrets"
echo "2. Created a secret in AWS Secrets Manager with secure generation"
echo "3. Added a least-privilege resource policy to control access to the secret"
echo "4. Retrieved the secret value (simulating application access)"
echo "5. Updated the secret with cryptographically secure values"
```

```
echo "6. Verified the updated secret"
echo ""
echo "Security best practices applied:"
echo "- Used cryptographically secure random ID generation"
echo "- Applied least-privilege resource policies with version stages"
echo "- Tagged resources for rotation tracking"
echo "- Blocked public access to secrets"
echo "- Used ARN-specific permissions instead of wildcards"
echo ""
echo "Next steps you might want to consider:"
echo "- Enable automatic secret rotation with AWS Lambda"
echo "- Implement secret caching in your application"
echo "- Set up CloudTrail logging for secret access auditing"
echo "- Use AWS CodeGuru Reviewer to find hardcoded secrets in your code"
echo "- For multi-region applications, replicate your secrets across regions"
echo "- Configure VPC endpoints for private access to Secrets Manager"
echo ""

echo "Script completed at $(date)"
exit 0
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateRole
  - CreateSecret
  - DeleteRole
  - DeleteSecret
  - DetachRolePolicy
  - GetCallerIdentity
  - GetSecretValue
  - PutResourcePolicy
  - UpdateSecret

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 FIS 在 EC2 实例上运行 CPU 压力测试

以下代码示例展示了如何：

- 创建 IAM 角色
- 创建 CloudWatch 告警
- 创建实验模板
- 运行试验
- 验证结果
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS FIS CPU Stress Test Tutorial Script
# This script automates the steps in the AWS FIS CPU stress test tutorial

#    approach using epoch time calculations that work across all Linux
 distributions

# Set up logging
LOG_FILE="fis-tutorial-$(date +%Y%m%d-%H%M%S).log"
exec > >(tee -a "$LOG_FILE") 2>&1

echo "Starting AWS FIS CPU Stress Test Tutorial Script"
echo "Logging to $LOG_FILE"
echo "================================================"

# Function to check for errors in command output
check_error() {
```

```
    local output=$1
    local cmd=$2

    if echo "$output" | grep -i "error" > /dev/null; then
        # Ignore specific expected errors
        if [[ "$cmd" == *"aws fis get-experiment"* ]] && [[ "$output" ==
 *"ConfigurationFailure"* ]]; then
            echo "Note: Experiment failed due to configuration issue. This is
 expected in some cases."
            return 0
        fi

        echo "ERROR: Command failed: $cmd"
        echo "Output: $output"
        cleanup_on_error
        exit 1
    fi
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "Error encountered. Cleaning up resources..."

    if [ -n "$EXPERIMENT_ID" ]; then
        echo "Stopping experiment $EXPERIMENT_ID if running..."
        aws fis stop-experiment --id "$EXPERIMENT_ID" 2>/dev/null || true
    fi

    if [ -n "$TEMPLATE_ID" ]; then
        echo "Deleting experiment template $TEMPLATE_ID..."
        aws fis delete-experiment-template --id "$TEMPLATE_ID" || true
    fi

    if [ -n "$INSTANCE_ID" ]; then
        echo "Terminating EC2 instance $INSTANCE_ID..."
        aws ec2 terminate-instances --instance-ids "$INSTANCE_ID" || true
    fi

    if [ -n "$ALARM_NAME" ]; then
        echo "Deleting CloudWatch alarm $ALARM_NAME..."
        aws cloudwatch delete-alarms --alarm-names "$ALARM_NAME" || true
    fi

    if [ -n "$INSTANCE_PROFILE_NAME" ]; then
```

```
        echo "Removing role from instance profile..."
        aws iam remove-role-from-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" --role-name "$EC2_ROLE_NAME" || true

        echo "Deleting instance profile..."
        aws iam delete-instance-profile --instance-profile-name
 "$INSTANCE_PROFILE_NAME" || true
    fi

    if [ -n "$FIS_ROLE_NAME" ]; then
        echo "Deleting FIS role policy..."
        aws iam delete-role-policy --role-name "$FIS_ROLE_NAME" --policy-name
 "$FIS_POLICY_NAME" || true

        echo "Deleting FIS role..."
        aws iam delete-role --role-name "$FIS_ROLE_NAME" || true
    fi

    if [ -n "$EC2_ROLE_NAME" ]; then
        echo "Detaching policy from EC2 role..."
        aws iam detach-role-policy --role-name "$EC2_ROLE_NAME" --policy-arn
 "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore" || true

        echo "Deleting EC2 role..."
        aws iam delete-role --role-name "$EC2_ROLE_NAME" || true
    fi

    echo "Cleanup completed."
}

# Generate unique identifiers for resources
TIMESTAMP=$(date +%Y%m%d%H%M%S)
FIS_ROLE_NAME="FISRole-${TIMESTAMP}"
FIS_POLICY_NAME="FISPolicy-${TIMESTAMP}"
EC2_ROLE_NAME="EC2SSMRole-${TIMESTAMP}"
INSTANCE_PROFILE_NAME="EC2SSMProfile-${TIMESTAMP}"
ALARM_NAME="FIS-CPU-Alarm-${TIMESTAMP}"

# Track created resources
CREATED_RESOURCES=()

echo "Step 1: Creating IAM role for AWS FIS"
# Create trust policy file for AWS FIS
cat > fis-trust-policy.json << 'EOF'
```

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "fis.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for FIS
echo "Creating IAM role $FIS_ROLE_NAME for AWS FIS..."
FIS_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$FIS_ROLE_NAME" \
  --assume-role-policy-document file://fis-trust-policy.json)
check_error "$FIS_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$FIS_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $FIS_ROLE_NAME")

# Create policy document for SSM actions
cat > fis-ssm-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ssm:SendCommand",
        "ssm:ListCommands",
        "ssm:ListCommandInvocations"
      ],
      "Resource": "*"
    }
  ]
}
EOF

# Attach policy to the role
echo "Attaching policy $FIS_POLICY_NAME to role $FIS_ROLE_NAME..."
```

```
FIS_POLICY_OUTPUT=$(aws iam put-role-policy \
  --role-name "$FIS_ROLE_NAME" \
  --policy-name "$FIS_POLICY_NAME" \
  --policy-document file://fis-ssm-policy.json)
check_error "$FIS_POLICY_OUTPUT" "aws iam put-role-policy"
CREATED_RESOURCES+=("IAM Policy: $FIS_POLICY_NAME attached to $FIS_ROLE_NAME")

echo "Step 2: Creating IAM role for EC2 instance with SSM permissions"
# Create trust policy file for EC2
cat > ec2-trust-policy.json << 'EOF'
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "ec2.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
EOF

# Create IAM role for EC2
echo "Creating IAM role $EC2_ROLE_NAME for EC2 instance..."
EC2_ROLE_OUTPUT=$(aws iam create-role \
  --role-name "$EC2_ROLE_NAME" \
  --assume-role-policy-document file://ec2-trust-policy.json)
check_error "$EC2_ROLE_OUTPUT" "aws iam create-role"
aws iam tag-role --role-name "$EC2_ROLE_NAME" --tags Key=project,Value=doc-smith
 Key=tutorial,Value=aws-fault-injection-service-gs
CREATED_RESOURCES+=("IAM Role: $EC2_ROLE_NAME")

# Attach SSM policy to the EC2 role
echo "Attaching AmazonSSMManagedInstanceCore policy to role $EC2_ROLE_NAME..."
EC2_POLICY_OUTPUT=$(aws iam attach-role-policy \
  --role-name "$EC2_ROLE_NAME" \
  --policy-arn arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore)
check_error "$EC2_POLICY_OUTPUT" "aws iam attach-role-policy"
CREATED_RESOURCES+=("IAM Policy: AmazonSSMManagedInstanceCore attached to
 $EC2_ROLE_NAME")

# Create instance profile
```

```
echo "Creating instance profile $INSTANCE_PROFILE_NAME..."
PROFILE_OUTPUT=$(aws iam create-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME")
check_error "$PROFILE_OUTPUT" "aws iam create-instance-profile"
CREATED_RESOURCES+=("IAM Instance Profile: $INSTANCE_PROFILE_NAME")

# Add role to instance profile
echo "Adding role $EC2_ROLE_NAME to instance profile $INSTANCE_PROFILE_NAME..."
ADD_ROLE_OUTPUT=$(aws iam add-role-to-instance-profile \
  --instance-profile-name "$INSTANCE_PROFILE_NAME" \
  --role-name "$EC2_ROLE_NAME")
check_error "$ADD_ROLE_OUTPUT" "aws iam add-role-to-instance-profile"

# Wait for role to propagate
echo "Waiting for IAM role to propagate..."
sleep 10

echo "Step 3: Launching EC2 instance"
# Get the latest Amazon Linux 2 AMI ID
echo "Finding latest Amazon Linux 2 AMI..."
AMI_ID=$(aws ec2 describe-images \
  --owners amazon \
  --filters "Name=name,Values=amzn2-ami-hvm-*-x86_64-gp2"
 "Name=state,Values=available" \
  --query "sort_by(Images, &CreationDate)[-1].ImageId" \
  --output text)
check_error "$AMI_ID" "aws ec2 describe-images"
echo "Using AMI: $AMI_ID"

# Launch EC2 instance
echo "Launching EC2 instance with AMI $AMI_ID..."
INSTANCE_OUTPUT=$(aws ec2 run-instances \
  --image-id "$AMI_ID" \
  --instance-type t2.micro \
  --iam-instance-profile Name="$INSTANCE_PROFILE_NAME" \
  --tag-specifications 'ResourceType=instance,Tags=[{Key=Name,Value=FIS-Test-
Instance},{Key=project,Value=doc-smith},{Key=tutorial,Value=aws-fault-injection-
service-gs}]')
check_error "$INSTANCE_OUTPUT" "aws ec2 run-instances"

# Get instance ID
INSTANCE_ID=$(echo "$INSTANCE_OUTPUT" | grep -i "InstanceId" | head -1 | awk -
F'"' '{print $4}')
if [ -z "$INSTANCE_ID" ]; then
```

```
    echo "Failed to get instance ID"
    cleanup_on_error
    exit 1
fi
echo "Launched instance: $INSTANCE_ID"
CREATED_RESOURCES+=("EC2 Instance: $INSTANCE_ID")

# Enable detailed monitoring
echo "Enabling detailed monitoring for instance $INSTANCE_ID..."
MONITOR_OUTPUT=$(aws ec2 monitor-instances --instance-ids "$INSTANCE_ID")
check_error "$MONITOR_OUTPUT" "aws ec2 monitor-instances"

# Wait for instance to be running and status checks to pass
echo "Waiting for instance to be ready..."
aws ec2 wait instance-running --instance-ids "$INSTANCE_ID"
aws ec2 wait instance-status-ok --instance-ids "$INSTANCE_ID"
echo "Instance is ready"

echo "Step 4: Creating CloudWatch alarm for CPU utilization"
# Create CloudWatch alarm
echo "Creating CloudWatch alarm $ALARM_NAME..."
ALARM_OUTPUT=$(aws cloudwatch put-metric-alarm \
  --alarm-name "$ALARM_NAME" \
  --alarm-description "Alarm when CPU exceeds 50%" \
  --metric-name CPUUtilization \
  --namespace AWS/EC2 \
  --statistic Maximum \
  --period 60 \
  --threshold 50 \
  --comparison-operator GreaterThanOrEqualToThreshold \
  --dimensions "Name=InstanceId,Value=$INSTANCE_ID" \
  --evaluation-periods 1)
check_error "$ALARM_OUTPUT" "aws cloudwatch put-metric-alarm"
CREATED_RESOURCES+=("CloudWatch Alarm: $ALARM_NAME")

# Get the alarm ARN
echo "Getting CloudWatch alarm ARN..."
ALARM_ARN_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
check_error "$ALARM_ARN_OUTPUT" "aws cloudwatch describe-alarms"
ALARM_ARN=$(echo "$ALARM_ARN_OUTPUT" | grep -i "AlarmArn" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$ALARM_ARN" ]; then
    echo "Failed to get alarm ARN"
```

```
    cleanup_on_error
    exit 1
fi
echo "Alarm ARN: $ALARM_ARN"

# Wait for the alarm to initialize and reach OK state
echo "Waiting for CloudWatch alarm to initialize (60 seconds)..."
sleep 60

# Check alarm state
echo "Checking alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
  --alarm-names "$ALARM_NAME")
ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 | awk -
F'"' '{print $4}')
echo "Current alarm state: $ALARM_STATE"

# If alarm is not in OK state, wait longer or generate some baseline metrics
if [ "$ALARM_STATE" != "OK" ]; then
    echo "Alarm not in OK state. Waiting for alarm to stabilize (additional 60
 seconds)..."
    sleep 60

    # Check alarm state again
    ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms \
      --alarm-names "$ALARM_NAME")
    ALARM_STATE=$(echo "$ALARM_STATE_OUTPUT" | grep -i "StateValue" | head -1 |
 awk -F'"' '{print $4}')
    echo "Updated alarm state: $ALARM_STATE"

    if [ "$ALARM_STATE" != "OK" ]; then
        echo "Warning: Alarm still not in OK state. Experiment may fail to
 start."
    fi
fi

echo "Step 5: Creating AWS FIS experiment template"
# Get the IAM role ARN
echo "Getting IAM role ARN for $FIS_ROLE_NAME..."
ROLE_ARN_OUTPUT=$(aws iam get-role \
  --role-name "$FIS_ROLE_NAME")
check_error "$ROLE_ARN_OUTPUT" "aws iam get-role"
ROLE_ARN=$(echo "$ROLE_ARN_OUTPUT" | grep -i "Arn" | head -1 | awk -F'"' '{print
 $4}')
```

```
if [ -z "$ROLE_ARN" ]; then
    echo "Failed to get role ARN"
    cleanup_on_error
    exit 1
fi
echo "Role ARN: $ROLE_ARN"

# Get account ID and region
ACCOUNT_ID=$(aws sts get-caller-identity --query "Account" --output text)
REGION=$(aws configure get region)
if [ -z "$REGION" ]; then
    REGION="us-east-1"  # Default to us-east-1 if region not set
fi
INSTANCE_ARN="arn:aws:ec2:${REGION}:${ACCOUNT_ID}:instance/${INSTANCE_ID}"
echo "Instance ARN: $INSTANCE_ARN"

# Create experiment template - Fixed JSON escaping issue
cat > experiment-template.json << EOF
{
  "description": "Test CPU stress predefined SSM document",
  "targets": {
    "testInstance": {
      "resourceType": "aws:ec2:instance",
      "resourceArns": ["$INSTANCE_ARN"],
      "selectionMode": "ALL"
    }
  },
  "actions": {
    "runCpuStress": {
      "actionId": "aws:ssm:send-command",
      "parameters": {
        "documentArn": "arn:aws:ssm:$REGION::document/AWSFIS-Run-CPU-Stress",
        "documentParameters": "{\"DurationSeconds\":\"120\"}",
        "duration": "PT5M"
      },
      "targets": {
        "Instances": "testInstance"
      }
    }
  },
  "stopConditions": [
    {
      "source": "aws:cloudwatch:alarm",
      "value": "$ALARM_ARN"
```

```
    }
  ],
  "roleArn": "$ROLE_ARN",
  "tags": {
    "Name": "FIS-CPU-Stress-Experiment",
    "project": "doc-smith",
    "tutorial": "aws-fault-injection-service-gs"
  }
}
EOF

# Create experiment template
echo "Creating AWS FIS experiment template..."
TEMPLATE_OUTPUT=$(aws fis create-experiment-template --cli-input-json file://
experiment-template.json)
check_error "$TEMPLATE_OUTPUT" "aws fis create-experiment-template"
TEMPLATE_ID=$(echo "$TEMPLATE_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$TEMPLATE_ID" ]; then
    echo "Failed to get template ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment template created with ID: $TEMPLATE_ID"
CREATED_RESOURCES+=("FIS Experiment Template: $TEMPLATE_ID")

echo "Step 6: Starting the experiment"
# Start the experiment
echo "Starting AWS FIS experiment using template $TEMPLATE_ID..."
EXPERIMENT_OUTPUT=$(aws fis start-experiment \
  --experiment-template-id "$TEMPLATE_ID" \
  --tags '{"Name": "FIS-CPU-Stress-Run"}')
check_error "$EXPERIMENT_OUTPUT" "aws fis start-experiment"
EXPERIMENT_ID=$(echo "$EXPERIMENT_OUTPUT" | grep -i "id" | head -1 | awk -F'"'
 '{print $4}')
if [ -z "$EXPERIMENT_ID" ]; then
    echo "Failed to get experiment ID"
    cleanup_on_error
    exit 1
fi
echo "Experiment started with ID: $EXPERIMENT_ID"
CREATED_RESOURCES+=("FIS Experiment: $EXPERIMENT_ID")

echo "Step 7: Tracking experiment progress"
```

```
# Track experiment progress
echo "Tracking experiment progress..."
MAX_CHECKS=30
CHECK_COUNT=0
EXPERIMENT_STATE=""

while [ $CHECK_COUNT -lt $MAX_CHECKS ]; do
    EXPERIMENT_INFO=$(aws fis get-experiment --id "$EXPERIMENT_ID")
    # Don't check for errors here, as we expect some experiments to fail

    EXPERIMENT_STATE=$(echo "$EXPERIMENT_INFO" | grep -i "status" | head -1 | awk
 -F'"' '{print $4}')
    echo "Experiment state: $EXPERIMENT_STATE"

    if [ "$EXPERIMENT_STATE" == "completed" ] || [ "$EXPERIMENT_STATE" ==
 "stopped" ] || [ "$EXPERIMENT_STATE" == "failed" ]; then
        # Show the reason for the state
        REASON=$(echo "$EXPERIMENT_INFO" | grep -i "reason" | head -1 | awk -F'"'
 '{print $4}')
        if [ -n "$REASON" ]; then
            echo "Reason: $REASON"
        fi
        break
    fi

    echo "Waiting 10 seconds before checking again..."
    sleep 10
    CHECK_COUNT=$((CHECK_COUNT + 1))
done

if [ $CHECK_COUNT -eq $MAX_CHECKS ]; then
    echo "Experiment is taking longer than expected. You can check its status
 later using:"
    echo "aws fis get-experiment --id $EXPERIMENT_ID"
fi

echo "Step 8: Verifying experiment results"
# Check CloudWatch alarm state
echo "Checking CloudWatch alarm state..."
ALARM_STATE_OUTPUT=$(aws cloudwatch describe-alarms --alarm-names "$ALARM_NAME")
check_error "$ALARM_STATE_OUTPUT" "aws cloudwatch describe-alarms"
echo "$ALARM_STATE_OUTPUT"

# Get CPU utilization metrics
```

```
echo "Getting CPU utilization metrics..."
END_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")

# FIXED: Cross-platform compatible way to calculate time 10 minutes ago
# This approach uses epoch seconds and basic arithmetic which works on all Linux
 distributions
CURRENT_EPOCH=$(date +%s)
TEN_MINUTES_AGO_EPOCH=$((CURRENT_EPOCH - 600))
START_TIME=$(date -u -d "@$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ" 2>/dev/
null || date -u -r "$TEN_MINUTES_AGO_EPOCH" +"%Y-%m-%dT%H:%M:%SZ")

# Create metric query file
cat > metric-query.json << EOF
[
  {
    "Id": "cpu",
    "MetricStat": {
      "Metric": {
        "Namespace": "AWS/EC2",
        "MetricName": "CPUUtilization",
        "Dimensions": [
          {
            "Name": "InstanceId",
            "Value": "$INSTANCE_ID"
          }
        ]
      },
      "Period": 60,
      "Stat": "Maximum"
    }
  }
]
EOF

METRICS_OUTPUT=$(aws cloudwatch get-metric-data \
  --start-time "$START_TIME" \
  --end-time "$END_TIME" \
  --metric-data-queries file://metric-query.json)
check_error "$METRICS_OUTPUT" "aws cloudwatch get-metric-data"
echo "CPU Utilization Metrics:"
echo "$METRICS_OUTPUT"

# Display summary of created resources
echo ""
```

```
echo "============================================="
echo "RESOURCES CREATED"
echo "============================================="
for resource in "${CREATED_RESOURCES[@]}"; do
    echo "- $resource"
done
echo "============================================="

# Prompt for cleanup
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
read -r CLEANUP_CHOICE

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Starting cleanup process..."

    # Stop experiment if still running
    if [ "$EXPERIMENT_STATE" != "completed" ] && [ "$EXPERIMENT_STATE" !=
 "stopped" ] && [ "$EXPERIMENT_STATE" != "failed" ]; then
        echo "Stopping experiment $EXPERIMENT_ID..."
        STOP_OUTPUT=$(aws fis stop-experiment --id "$EXPERIMENT_ID")
        check_error "$STOP_OUTPUT" "aws fis stop-experiment"
        echo "Waiting for experiment to stop..."
        sleep 10
    fi

    # Delete experiment template
    echo "Deleting experiment template $TEMPLATE_ID..."
    DELETE_TEMPLATE_OUTPUT=$(aws fis delete-experiment-template --id
 "$TEMPLATE_ID")
    check_error "$DELETE_TEMPLATE_OUTPUT" "aws fis delete-experiment-template"

    # Delete CloudWatch alarm
    echo "Deleting CloudWatch alarm $ALARM_NAME..."
    DELETE_ALARM_OUTPUT=$(aws cloudwatch delete-alarms --alarm-names
 "$ALARM_NAME")
    check_error "$DELETE_ALARM_OUTPUT" "aws cloudwatch delete-alarms"

    # Terminate EC2 instance
    echo "Terminating EC2 instance $INSTANCE_ID..."
    TERMINATE_OUTPUT=$(aws ec2 terminate-instances --instance-ids "$INSTANCE_ID")
```

```
    check_error "$TERMINATE_OUTPUT" "aws ec2 terminate-instances"
    echo "Waiting for instance to terminate..."
    aws ec2 wait instance-terminated --instance-ids "$INSTANCE_ID"

    # Clean up IAM resources
    echo "Removing role from instance profile..."
    REMOVE_ROLE_OUTPUT=$(aws iam remove-role-from-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME" \
      --role-name "$EC2_ROLE_NAME")
    check_error "$REMOVE_ROLE_OUTPUT" "aws iam remove-role-from-instance-profile"

    echo "Deleting instance profile..."
    DELETE_PROFILE_OUTPUT=$(aws iam delete-instance-profile \
      --instance-profile-name "$INSTANCE_PROFILE_NAME")
    check_error "$DELETE_PROFILE_OUTPUT" "aws iam delete-instance-profile"

    echo "Deleting FIS role policy..."
    DELETE_POLICY_OUTPUT=$(aws iam delete-role-policy \
      --role-name "$FIS_ROLE_NAME" \
      --policy-name "$FIS_POLICY_NAME")
    check_error "$DELETE_POLICY_OUTPUT" "aws iam delete-role-policy"

    echo "Detaching policy from EC2 role..."
    DETACH_POLICY_OUTPUT=$(aws iam detach-role-policy \
      --role-name "$EC2_ROLE_NAME" \
      --policy-arn "arn:aws:iam::aws:policy/AmazonSSMManagedInstanceCore")
    check_error "$DETACH_POLICY_OUTPUT" "aws iam detach-role-policy"

    echo "Deleting FIS role..."
    DELETE_FIS_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$FIS_ROLE_NAME")
    check_error "$DELETE_FIS_ROLE_OUTPUT" "aws iam delete-role"

    echo "Deleting EC2 role..."
    DELETE_EC2_ROLE_OUTPUT=$(aws iam delete-role \
      --role-name "$EC2_ROLE_NAME")
    check_error "$DELETE_EC2_ROLE_OUTPUT" "aws iam delete-role"

    # Clean up temporary files
    echo "Cleaning up temporary files..."
    rm -f fis-trust-policy.json ec2-trust-policy.json fis-ssm-policy.json
experiment-template.json metric-query.json

    echo "Cleanup completed successfully."
```

```
else
    echo "Cleanup skipped. Resources will remain in your AWS account."
    echo "You can manually clean up the resources listed above."
fi

echo ""
echo "Script execution completed."
echo "Log file: $LOG_FILE"
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AddRoleToInstanceProfile
  - AttachRolePolicy
  - CreateExperimentTemplate
  - CreateInstanceProfile
  - CreateRole
  - DeleteAlarms
  - DeleteExperimentTemplate
  - DeleteInstanceProfile
  - DeleteRole
  - DeleteRolePolicy
  - DescribeAlarms
  - DescribeImages
  - DetachRolePolicy
  - GetCallerIdentity
  - GetExperiment
  - GetMetricData
  - GetRole
  - MonitorInstances
  - PutMetricAlarm
  - PutRolePolicy
  - RemoveRoleFromInstanceProfile
  - RunInstances

- StartExperiment
- StopExperiment
- TerminateInstances
- Wait

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

设置 Systems Manager

以下代码示例展示了如何：

- 为 Systems Manager 创建 IAM 权限
- 为 Systems Manager 创建 IAM 角色
- 配置 Systems Manager
- 验证设置
- 清理资源

Bash

Amazon CLI 及 Bash 脚本

> **Note**
>
> 查看 GitHub，了解更多信息。在 Sample developer tutorials 存储库中查找完整示例，了解如何进行设置和运行。

```
#!/bin/bash

# AWS Systems Manager Setup Script
# This script sets up AWS Systems Manager for a single account and region
#
# Version 17 fixes:
# 1. Added cloudformation.amazonaws.com to the IAM role trust policy
# 2. Systems Manager Quick Setup uses CloudFormation for deployments, so the role
 must trust CloudFormation service
```

```
# Initialize log file
LOG_FILE="ssm_setup_$(date +%Y%m%d_%H%M%S).log"
UNIQUE_ID=$(openssl rand -hex 4)
echo "Starting AWS Systems Manager setup at $(date)" > "$LOG_FILE"

# Function to log commands and their outputs with immediate terminal display
log_cmd() {
    echo "$(date): Running command: $1" | tee -a "$LOG_FILE"
    local output
    output=$(eval "$1" 2>&1)
    local status=$?
    echo "$output" | tee -a "$LOG_FILE"
    return $status
}

# Function to check for errors in command output
check_error() {
    local cmd_output="$1"
    local cmd_status="$2"
    local error_msg="$3"

    if [[ $cmd_status -ne 0 || "$cmd_output" =~ [Ee][Rr][Rr][Oo][Rr] ]]; then
        echo "ERROR: $error_msg" | tee -a "$LOG_FILE"
        echo "Command output: $cmd_output" | tee -a "$LOG_FILE"
        cleanup_on_error
        exit 1
    fi
}

# Array to track created resources for cleanup
declare -a CREATED_RESOURCES

# Function to add a resource to the tracking array
track_resource() {
    local resource_type="$1"
    local resource_id="$2"
    CREATED_RESOURCES+=("$resource_type:$resource_id")
    echo "Tracked resource: $resource_type:$resource_id" | tee -a "$LOG_FILE"
}

# Function to clean up resources on error
cleanup_on_error() {
    echo "" | tee -a "$LOG_FILE"
```

```
    echo "========================================" | tee -a "$LOG_FILE"
    echo "ERROR OCCURRED - CLEANING UP RESOURCES" | tee -a "$LOG_FILE"
    echo "========================================" | tee -a "$LOG_FILE"
    echo "The following resources were created:" | tee -a "$LOG_FILE"

    # Display resources in reverse order
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        echo "${CREATED_RESOURCES[$i]}" | tee -a "$LOG_FILE"
    done

    echo "" | tee -a "$LOG_FILE"
    echo "Attempting to clean up resources..." | tee -a "$LOG_FILE"

    # Clean up resources in reverse order
    cleanup_resources
}

# Function to clean up all created resources
cleanup_resources() {
    # Process resources in reverse order (last created, first deleted)
    for ((i=${#CREATED_RESOURCES[@]}-1; i>=0; i--)); do
        IFS=':' read -r resource_type resource_id <<< "${CREATED_RESOURCES[$i]}"

        echo "Deleting $resource_type: $resource_id" | tee -a "$LOG_FILE"

        case "$resource_type" in
            "IAM_POLICY")
                # Delete the policy (detachment should have been handled when the
 role was deleted)
                log_cmd "aws iam delete-policy --policy-arn $resource_id" || true
                ;;
            "IAM_ROLE")
                # Detach all policies from the role first
                if [[ -n "$POLICY_ARN" ]]; then
                    log_cmd "aws iam detach-role-policy --role-name $resource_id
 --policy-arn $POLICY_ARN" || true
                fi

                # Delete the role
                log_cmd "aws iam delete-role --role-name $resource_id" || true
                ;;
            "SSM_CONFIG_MANAGER")
                log_cmd "aws ssm-quicksetup delete-configuration-manager --
manager-arn $resource_id" || true
```

```
                ;;
            *)
                echo "Unknown resource type: $resource_type, cannot delete
 automatically" | tee -a "$LOG_FILE"
                ;;
        esac
    done

    echo "Cleanup completed" | tee -a "$LOG_FILE"

    # Clean up temporary files
    rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/
null || true
}

# Main script execution
echo "AWS Systems Manager Setup Script"
echo "================================"
echo "This script will set up AWS Systems Manager for a single account and
 region."
echo "It will create IAM policies and roles, then enable Systems Manager
 features."
echo ""

# Get the current AWS region
CURRENT_REGION=$(aws configure get region)
if [[ -z "$CURRENT_REGION" ]]; then
    echo "No AWS region configured. Please specify a region:"
    CURRENT_REGION="${AWS_DEFAULT_REGION:-us-west-2}"
    if [[ -z "$CURRENT_REGION" ]]; then
        echo "ERROR: A region must be specified" | tee -a "$LOG_FILE"
        exit 1
    fi
fi

echo "Using AWS region: $CURRENT_REGION" | tee -a "$LOG_FILE"

# Step 1: Create IAM policy for Systems Manager onboarding
echo "Step 1: Creating IAM policy for Systems Manager onboarding..."

# Create policy document
cat > ssm-onboarding-policy.json << 'EOF'
{
   "Version":"2012-10-17",
```

```
    "Statement": [
      {
        "Sid": "QuickSetupActions",
        "Effect": "Allow",
        "Action": [
          "ssm-quicksetup:*"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmReadOnly",
        "Effect": "Allow",
        "Action": [
          "ssm:DescribeAutomationExecutions",
          "ssm:GetAutomationExecution",
          "ssm:ListAssociations",
          "ssm:DescribeAssociation",
          "ssm:ListDocuments",
          "ssm:ListResourceDataSync",
          "ssm:DescribePatchBaselines",
          "ssm:GetPatchBaseline",
          "ssm:DescribeMaintenanceWindows",
          "ssm:DescribeMaintenanceWindowTasks"
        ],
        "Resource": "*"
      },
      {
        "Sid": "SsmDocument",
        "Effect": "Allow",
        "Action": [
          "ssm:GetDocument",
          "ssm:DescribeDocument"
        ],
        "Resource": [
          "arn:aws:ssm:*:*:document/AWSQuickSetupType-*",
          "arn:aws:ssm:*:*:document/AWS-EnableExplorer"
        ]
      },
      {
        "Sid": "SsmEnableExplorer",
        "Effect": "Allow",
        "Action": "ssm:StartAutomationExecution",
        "Resource": "arn:aws:ssm:*:*:automation-definition/AWS-EnableExplorer:*"
      },
```

```
    {
      "Sid": "SsmExplorerRds",
      "Effect": "Allow",
      "Action": [
        "ssm:GetOpsSummary",
        "ssm:CreateResourceDataSync",
        "ssm:UpdateResourceDataSync"
      ],
      "Resource": "arn:aws:ssm:*:*:resource-data-sync/AWS-QuickSetup-*"
    },
    {
      "Sid": "OrgsReadOnly",
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListRoots",
        "organizations:ListParents",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAWSServiceAccessForOrganization"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgsAdministration",
      "Effect": "Allow",
      "Action": [
        "organizations:EnableAWSServiceAccess",
        "organizations:RegisterDelegatedAdministrator",
        "organizations:DeregisterDelegatedAdministrator"
      ],
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "organizations:ServicePrincipal": [
            "ssm.amazonaws.com",
            "ssm-quicksetup.amazonaws.com",
            "member.org.stacksets.cloudformation.amazonaws.com",
            "resource-explorer-2.amazonaws.com"
          ]
        }
      }
```

```
    },
    {
      "Sid": "CfnReadOnly",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ListStacks",
        "cloudformation:DescribeStacks",
        "cloudformation:ListStackSets",
        "cloudformation:DescribeOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "OrgCfnAccess",
      "Effect": "Allow",
      "Action": [
        "cloudformation:ActivateOrganizationsAccess"
      ],
      "Resource": "*"
    },
    {
      "Sid": "CfnStackActions",
      "Effect": "Allow",
      "Action": [
        "cloudformation:CreateStack",
        "cloudformation:DeleteStack",
        "cloudformation:DescribeStackResources",
        "cloudformation:DescribeStackEvents",
        "cloudformation:GetTemplate",
        "cloudformation:RollbackStack",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStack"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stack/StackSet-AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:stack/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*"
      ]
    },
    {
      "Sid": "CfnStackSetActions",
      "Effect": "Allow",
      "Action": [
```

```
        "cloudformation:CreateStackInstances",
        "cloudformation:CreateStackSet",
        "cloudformation:DeleteStackInstances",
        "cloudformation:DeleteStackSet",
        "cloudformation:DescribeStackInstance",
        "cloudformation:DetectStackSetDrift",
        "cloudformation:ListStackInstanceResourceDrifts",
        "cloudformation:DescribeStackSet",
        "cloudformation:DescribeStackSetOperation",
        "cloudformation:ListStackInstances",
        "cloudformation:ListStackSetOperations",
        "cloudformation:ListStackSetOperationResults",
        "cloudformation:TagResource",
        "cloudformation:UntagResource",
        "cloudformation:UpdateStackSet"
      ],
      "Resource": [
        "arn:aws:cloudformation:*:*:stackset/AWS-QuickSetup-*",
        "arn:aws:cloudformation:*:*:type/resource/*",
        "arn:aws:cloudformation:*:*:stackset-target/AWS-QuickSetup-*:*"
      ]
    },
    {
      "Sid": "ValidationReadonlyActions",
      "Effect": "Allow",
      "Action": [
        "iam:ListRoles",
        "iam:GetRole"
      ],
      "Resource": "*"
    },
    {
      "Sid": "IamRolesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:CreateRole",
        "iam:DeleteRole",
        "iam:GetRole",
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy",
        "iam:GetRolePolicy",
        "iam:ListRolePolicies"
      ],
      "Resource": [
```

```
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ]
    },
    {
      "Sid": "IamPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:PassRole"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "StringEquals": {
          "iam:PassedToService": [
            "ssm.amazonaws.com",
            "ssm-quicksetup.amazonaws.com",
            "cloudformation.amazonaws.com"
          ]
        }
      }
    },
    {
      "Sid": "IamRolesPoliciesMgmt",
      "Effect": "Allow",
      "Action": [
        "iam:AttachRolePolicy",
        "iam:DetachRolePolicy"
      ],
      "Resource": [
        "arn:aws:iam::*:role/AWS-QuickSetup-*",
        "arn:aws:iam::*:role/service-role/AWS-QuickSetup-*"
      ],
      "Condition": {
        "ArnEquals": {
          "iam:PolicyARN": [
            "arn:aws:iam::aws:policy/
AWSSystemsManagerEnableExplorerExecutionPolicy",
            "arn:aws:iam::aws:policy/AWSQuickSetupSSMDeploymentRolePolicy"
          ]
        }
      }
```

```
     },
     {
       "Sid": "CfnStackSetsSLR",
       "Effect": "Allow",
       "Action": [
         "iam:CreateServiceLinkedRole"
       ],
       "Resource": [
         "arn:aws:iam::*:role/aws-service-role/
stacksets.cloudformation.amazonaws.com/
AWSServiceRoleForCloudFormationStackSetsOrgAdmin",
         "arn:aws:iam::*:role/aws-service-role/ssm.amazonaws.com/
AWSServiceRoleForAmazonSSM",
         "arn:aws:iam::*:role/aws-service-role/
accountdiscovery.ssm.amazonaws.com/AWSServiceRoleForAmazonSSM_AccountDiscovery",
         "arn:aws:iam::*:role/aws-service-role/ssm-quicksetup.amazonaws.com/
AWSServiceRoleForSSMQuickSetup",
         "arn:aws:iam::*:role/aws-service-role/resource-explorer-2.amazonaws.com/
AWSServiceRoleForResourceExplorer"
       ]
     }
   ]
}
EOF

# Create the IAM policy
POLICY_OUTPUT=$(log_cmd "aws iam create-policy --policy-name SSMOnboardingPolicy-
$UNIQUE_ID --policy-document file://ssm-onboarding-policy.json --output json")
POLICY_STATUS=$?
check_error "$POLICY_OUTPUT" $POLICY_STATUS "Failed to create IAM policy"

# Extract the policy ARN
POLICY_ARN=$(echo "$POLICY_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:policy/
SSMOnboardingPolicy-[a-f0-9]*')
if [[ -z "$POLICY_ARN" ]]; then
    echo "ERROR: Failed to extract policy ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created policy
track_resource "IAM_POLICY" "$POLICY_ARN"
aws iam tag-policy --policy-arn "$POLICY_ARN" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs
```

```
echo "Created policy: $POLICY_ARN" | tee -a "$LOG_FILE"

# Step 2: Create and configure IAM role for Systems Manager
echo ""
echo "Step 2: Creating IAM role for Systems Manager..."

# Get current user name
USER_OUTPUT=$(log_cmd "aws sts get-caller-identity --output json")
USER_STATUS=$?
check_error "$USER_OUTPUT" $USER_STATUS "Failed to get caller identity"

# Extract account ID
ACCOUNT_ID=$(echo "$USER_OUTPUT" | grep -o '"Account": "[0-9]*"' | cut -d'"' -f4)
if [[ -z "$ACCOUNT_ID" ]]; then
    echo "ERROR: Failed to extract account ID" | tee -a "$LOG_FILE"
    exit 1
fi

# Generate a unique role name
ROLE_NAME="SSMTutorialRole-$(openssl rand -hex 4)"

# Create trust policy for the role - FIXED: Added cloudformation.amazonaws.com
cat > trust-policy.json << 'EOF'
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "ssm.amazonaws.com",
                    "ssm-quicksetup.amazonaws.com",
                    "cloudformation.amazonaws.com"
                ]
            },
            "Action": "sts:AssumeRole"
        },
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::ACCOUNT_ID:root"
            },
            "Action": "sts:AssumeRole"
        }
```

```
    ]
}
EOF

# Replace ACCOUNT_ID placeholder in trust policy
sed -i "s/ACCOUNT_ID/$ACCOUNT_ID/g" trust-policy.json

# Create the IAM role
ROLE_OUTPUT=$(log_cmd "aws iam create-role --role-name $ROLE_NAME --assume-role-
policy-document file://trust-policy.json --description 'Role for Systems Manager
 tutorial' --output json")
ROLE_STATUS=$?
check_error "$ROLE_OUTPUT" $ROLE_STATUS "Failed to create IAM role"

# Extract the role ARN
ROLE_ARN=$(echo "$ROLE_OUTPUT" | grep -o 'arn:aws:iam::[0-9]*:role/[^"]*')
if [[ -z "$ROLE_ARN" ]]; then
    echo "ERROR: Failed to extract role ARN" | tee -a "$LOG_FILE"
    cleanup_on_error
    exit 1
fi

# Track the created role
track_resource "IAM_ROLE" "$ROLE_NAME"
aws iam tag-role --role-name "$ROLE_NAME" \
  --tags Key=project,Value=doc-smith Key=tutorial,Value=aws-systems-manager-gs

echo "Created IAM role: $ROLE_NAME" | tee -a "$LOG_FILE"
echo "Role ARN: $ROLE_ARN" | tee -a "$LOG_FILE"

# Set identity variables for cleanup
IDENTITY_TYPE="role"
IDENTITY_NAME="$ROLE_NAME"

# Attach the policy to the role
ATTACH_OUTPUT=$(log_cmd "aws iam attach-role-policy --role-name $ROLE_NAME --
policy-arn $POLICY_ARN")
ATTACH_STATUS=$?
check_error "$ATTACH_OUTPUT" $ATTACH_STATUS "Failed to attach policy to role
 $ROLE_NAME"

echo "Policy attached to role: $ROLE_NAME" | tee -a "$LOG_FILE"

# Step 3: Create Systems Manager configuration using Host Management
```

```
echo ""
echo "Step 3: Creating Systems Manager configuration..."

# Generate a random identifier for the configuration name
CONFIG_NAME="SSMSetup-$(openssl rand -hex 4)"

# Create configuration file for Systems Manager setup using Host Management
# Added both required parameters for single account deployment based on
 CloudFormation documentation
cat > ssm-config.json << EOF
[
  {
    "Type": "AWSQuickSetupType-SSMHostMgmt",
    "LocalDeploymentAdministrationRoleArn": "$ROLE_ARN",
    "LocalDeploymentExecutionRoleName": "$ROLE_NAME",
    "Parameters": {
      "TargetAccounts": "$ACCOUNT_ID",
      "TargetRegions": "$CURRENT_REGION"
    }
  }
]
EOF

echo "Configuration file created:" | tee -a "$LOG_FILE"
cat ssm-config.json | tee -a "$LOG_FILE"

# Create the configuration manager
CONFIG_OUTPUT=$(log_cmd "aws ssm-quicksetup create-configuration-manager --name
 \"$CONFIG_NAME\" --configuration-definitions file://ssm-config.json --region
 $CURRENT_REGION")
CONFIG_STATUS=$?
check_error "$CONFIG_OUTPUT" $CONFIG_STATUS "Failed to create Systems Manager
 configuration"

# Extract the manager ARN
MANAGER_ARN=$(echo "$CONFIG_OUTPUT" | grep -o 'arn:aws:ssm-quicksetup:[^"]*')
if [[ -z "$MANAGER_ARN" ]]; then
    echo "ERROR: Failed to extract manager ARN" | tee -a "$LOG_FILE"
    exit 1
fi

# Track the created configuration manager
track_resource "SSM_CONFIG_MANAGER" "$MANAGER_ARN"
```

```
echo "Created Systems Manager configuration: $MANAGER_ARN" | tee -a "$LOG_FILE"

# Step 4: Verify the setup
echo ""
echo "Step 4: Verifying the setup..."

# Wait for the configuration to be fully deployed
echo "Waiting for the configuration to be deployed (this may take a few
 minutes)..."
sleep 30

# Check the configuration manager status
VERIFY_OUTPUT=$(log_cmd "aws ssm-quicksetup get-configuration-manager --manager-
arn $MANAGER_ARN --region $CURRENT_REGION")
VERIFY_STATUS=$?
check_error "$VERIFY_OUTPUT" $VERIFY_STATUS "Failed to verify configuration
 manager"

echo "Systems Manager setup completed successfully!" | tee -a "$LOG_FILE"

# List the created resources
echo ""
echo "============================================="
echo "CREATED RESOURCES"
echo "============================================="
for resource in "${CREATED_RESOURCES[@]+"${CREATED_RESOURCES[@]}"}"; do
    echo "$resource"
done

# Prompt for cleanup
echo ""
echo "============================================="
echo "CLEANUP CONFIRMATION"
echo "============================================="
echo "Do you want to clean up all created resources? (y/n): "
CLEANUP_CHOICE="y"

if [[ "$CLEANUP_CHOICE" =~ ^[Yy]$ ]]; then
    echo "Cleaning up resources..." | tee -a "$LOG_FILE"
    cleanup_resources
    echo "Cleanup completed." | tee -a "$LOG_FILE"
else
    echo "Resources will not be cleaned up. You can manually clean them up
 later." | tee -a "$LOG_FILE"
```

```
fi

echo ""
echo "Script execution completed. See $LOG_FILE for details."

# Clean up temporary files
rm -f ssm-onboarding-policy.json trust-policy.json ssm-config.json 2>/dev/null ||
 true
```

- 有关 API 详细信息，请参阅《Amazon CLI 命令参考》中的以下主题。
  - AttachRolePolicy
  - CreateConfigurationManager
  - CreatePolicy
  - CreateRole
  - DeleteConfigurationManager
  - DeletePolicy
  - DeleteRole
  - DetachRolePolicy
  - GetCallerIdentity
  - GetConfigurationManager

有关 Amazon SDK 开发人员指南和代码示例的完整列表，请参阅 将此服务与 Amazon SDK 结合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

IAM 和 Amazon STS 中的安全性

Amazon 的云安全性的优先级最高。为了满足对安全性最敏感的组织的需求，我们打造了具有超高安全性的数据中心和网络架构。作为 Amazon 的客户，您也可以从这些数据中心和网络架构受益。

安全性是 Amazon 和您的共同责任。责任共担模式将其描述为云的安全性和云中的安全性：

- 云的安全性：Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础结构。Amazon 还向您提供可安全使用的服务。第三方审核员定期测试和验证我们的安全性的有效性，作为 Amazon Compliance Programs 的一部分。要了解适用于 Amazon Identity and Access Management (IAM) 的合规性计划，请参阅合规性计划范围内的 Amazon 服务。
- 云中的安全性：您的责任由您使用的 Amazon 服务决定。您还需要对其它因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

此文档将帮助您了解如何在使用 Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (Amazon STS) 时应用责任共担模式。以下主题说明如何配置 IAM 和 Amazon STS 以实现您的安全性和合规性目标。您还会了解如何使用其他 Amazon 服务以帮助您监控和保护 IAM 资源。

内容

- Amazon 安全凭证
- Amazon 安全审核指南
- Amazon Identity and Access Management 中的数据保护
- Amazon Identity and Access Management 中的日志记录和监控
- Amazon Identity and Access Management 的合规性验证
- Amazon Identity and Access Management 中的故障恢复能力
- Amazon Identity and Access Management 中的基础结构安全性
- Amazon Identity and Access Management 中的配置和漏洞分析
- 适用于 Amazon Identity and Access Management and Access Analyzer 的 Amazon 托管式策略
- IAM 外部的安全功能

Amazon 安全凭证

当您与 Amazon 交互时，可指定 Amazon 安全凭证 以验证您的身份以及您是否有权访问所请求的资源。Amazon 使用安全凭证来对您的请求进行身份验证和授权。

例如，如果要从 Amazon Simple Storage Service (Amazon S3) 存储桶下载受保护的文件，则您的凭证必须允许该访问。如果您的凭证未显示您有权下载该文件，Amazon 会拒绝您的请求。但是，下载公开共享的 Amazon S3 存储桶中的文件不需要您的 Amazon 安全凭证。

Amazon 中有不同类型的用户，每种用户都有自己的安全凭证：

- 账户所有者（根用户）— 创建 Amazon Web Services 账户 并拥有完全访问权限的用户。
- Amazon IAM Identity Center 用户 — 在 Amazon IAM Identity Center 中管理的用户。
- 联合主体 – 来自外部身份提供者、通过联合身份验证获得 Amazon 临时访问权限的用户。有关联合身份的更多信息，请参阅 身份提供程序和 Amazon 中的联合身份验证。
- IAM 用户 — 在 Amazon Identity and Access Management（IAM）服务中创建的个人用户。

用户拥有长期或临时安全凭证。根用户、IAM 用户和访问密钥具有不会过期的长期安全凭证。为保护长期凭证，您可以制定相应的流程，以 管理访问密钥、更改密码 和 启用 MFA。

为了简化 Amazon Organizations 中跨成员账户的根用户凭证的管理，您可以集中保护使用 Amazon Organizations 管理的 Amazon Web Services 账户的根用户凭证。集中管理成员账户的根访问权限 允许您集中移除和防止长期的根用户凭证恢复，从而防止大规模的意外根访问。

IAM 角色、Amazon IAM Identity Center 中的用户 和 Amazon STS联合用户主体具有临时安全凭证。临时安全凭证在规定的时间段后或用户结束会话时过期。临时凭证的工作方式与长期凭证的工作方式几乎相同，仅存在以下差异：

- 顾名思义，临时安全凭证是短期 凭证。可将这些凭证的有效时间配置几分钟到几小时。一旦这些凭证到期，Amazon 将不再识别这些凭证或不再允许来自使用这些凭证发出的 API 请求的任何类型的访问。
- 临时安全凭证不随用户一起存储，而是动态生成并在用户提出请求时提供给用户。临时安全凭证到期时 (甚至之前)，用户可以请求新的凭证，只要请求凭证的用户仍有权这样做。

因此，与长期凭证相比，临时凭证具有以下优势：

- 您不必随应用程序分配或嵌入长期 Amazon 安全凭证。

- 可允许用户访问您的 Amazon 资源，而不必为这些用户定义 Amazon 身份。临时凭证是角色和联合身份验证的基础。
- 临时安全凭证的生命周期较短，因此无需更新或在不再需要这些凭证时显式撤销这些凭证。临时安全凭证到期后无法重复使用。您可指定凭证的有效期，但有最长限期。

安全性注意事项

在确定 Amazon Web Services 账户 的安全规定时，我们建议您考虑以下信息：

- 当您创建 Amazon Web Services 账户 时，我们会创建账户根用户。根用户（账户所有者）的凭证允许完全访问账户中的所有资源。您使用根用户执行的首项任务是向其他用户授予对您的 Amazon Web Services 账户 的管理权限，以便最大限度地减少根用户的使用。
- 多重身份验证 (MFA) 为可以访问 Amazon Web Services 账户 的用户提供了额外的安全级别。为了提高安全性，我们建议您要求对 Amazon Web Services 账户根用户 凭证和所有 IAM 用户使用 MFA。有关更多信息，请参阅 IAM 中的 Amazon 多重身份验证。
- Amazon 需要不同类型的安全凭证，具体取决于您访问 Amazon 的方式以及您所属的 Amazon 用户类型。例如，您可以使用登录凭证登入 Amazon Web Services 管理控制台，但对 Amazon 进行编程调用时则需要使用访问密钥。有关确定用户类型和登录页面的帮助，请参阅《Amazon 登录 用户指南》中的什么是 Amazon 登录。
- 您无法使用 IAM policy 显式拒绝根用户访问资源。您只能使用 Amazon Organizations 服务控制策略（SCP）来限制根用户的权限。
- 如果您忘记或丢失了根用户密码，则必须有权访问与您的账户关联的电子邮件地址才能重置根用户密码。
- 如果丢失了根用户访问密钥，则您必须能够以根用户身份登录您的账户才能创建新的访问密钥。
- 请不要将根用户用于您的日常任务。请使用它来执行仅限根用户可以执行的任务。有关需要您以根用户身份登录的任务的完整列表，请参阅 需要根用户凭证的任务。
- 安全凭证特定于账户。如果您有权访问多个 Amazon Web Services 账户，则每个账户都必须有单独的凭证。
- 策略确定用户、角色或用户组成员可以在什么样的条件下对哪些 Amazon 资源执行哪些操作。使用策略，您可以安全地控制对 Amazon Web Services 服务 和 Amazon Web Services 账户 中资源的访问。如果必须修改或撤销权限以响应安全事件，您可以删除或修改策略，而不是直接更改身份。
- 请务必将您的 Emergency Access IAM 用户的登录凭证以及您为编程访问创建的任何访问密钥保存在安全位置。如果您丢失了访问密钥，则必须登录您的账户才能创建新的访问密钥。

- 我们强烈建议您使用 IAM 角色和联合主体提供的临时凭证，而不是 IAM 用户和访问密钥提供的长期凭证。

使用 Amazon 安全凭证以编程方式进行访问

我们建议尽可能使用短期访问密钥来编程调用 Amazon 或使用 Amazon Command Line Interface 或 Amazon Tools for PowerShell。但是，您也可以将长期 Amazon 访问密钥用于这些目的。

创建长期访问密钥时，您将访问密钥 ID（例如 AKIAIOSFODNN7EXAMPLE）和秘密访问密钥（例如 wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY）创建为一组。秘密访问密钥仅在您创建它时可供下载。如果您没有下载秘密访问密钥或丢失了它，则必须创建新的秘密访问密钥。

在许多情况下，您并不需要永不过期的长期访问密钥（例如为 IAM 用户创建的访问密钥）。相反，您可以创建 IAM 角色并生成临时安全凭证。临时安全凭证包括访问密钥 ID 和秘密访问密钥，以及一个指示凭证何时到期的安全令牌。在过期后，这些凭证将不再有效。有关更多信息，请参阅 长期访问密钥的替代方案。

以 AKIA 开头的访问密钥 ID 是 IAM 用户或 Amazon Web Services 账户 根用户的长期访问密钥。以 ASIA 开头的访问密钥 ID 是使用 Amazon STS 操作创建的临时凭证访问密钥。

如果用户需要在 Amazon Web Services 管理控制台之外与 Amazon 交互，则需要编程式访问权限。Amazon API 和 Amazon Command Line Interface 需要访问密钥。可能的话，创建临时凭证，该凭证由一个访问密钥 ID、一个秘密访问密钥和一个指示凭证何时到期的安全令牌组成。

要向用户授予编程式访问权限，请选择以下选项之一。

哪个用户需要编程式访问权限？	目的	方式
IAM	使用短期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中将临时凭证用于 Amazon 资源中的说明进行操作。
IAM	（不推荐使用） 使用长期凭证签署对 Amazon CLI 或 Amazon API 的编程式请求（直接或使用 Amazon SDK）。	按照《IAM 用户指南》中管理 IAM 用户的访问密钥中的说明进行操作。

长期访问密钥的替代方案

对于许多常见使用案例，应有长期访问密钥的替代方法。为了提高您的账户安全性，请考虑以下几点。

- 不要在应用程序代码或代码存储库中嵌入长期访问密钥和秘密访问密钥 –改用 Amazon Secrets Manager 或其他秘密管理解决方案，因此您不必以纯文本硬编码密钥。然后，应用程序或客户端可以在需要时检索秘密。有关更多信息，请参阅《Amazon Secrets Manager 用户指南》中的什么是 Amazon Secrets Manager？。

- 尽可能使用 IAM 角色来生成临时安全凭证 –尽可能使用机制颁发临时安全凭证，而不是使用长期访问密钥。临时安全凭证更加安全，因为它们不随用户一起存储，而是动态生成并在用户提出请求时提供给用户。临时安全凭证的生命周期较短，因此无需管理或更新。提供临时访问密钥的机制包括 IAM 角色或 IAM Identity Center 用户的身份验证。对于在 Amazon 外部运行的计算机，您可以使用 Amazon Identity and Access Management Roles Anywhere。
- 对 Amazon Command Line Interface(Amazon CLI) 或 **`aws-shell`** 使用长期访问密钥的替代方法 – 替代方法包括如下。
  - 使用控制台凭证登录以进行 Amazon 本地开发。您可以使用 Amazon CLI 版本 2 和 `aws login` 命令生成短期凭证，以使用您的控制台凭证运行 Amazon CLI 命令。要了解更多信息，请参阅《Amazon Command Line Interface 用户指南》中的登录以进行 Amazon 本地开发。
  - Amazon CloudShell 是一个基于浏览器的预先验证 shell，您可以直接从 Amazon Web Services 管理控制台 中启动。您可以通过自己喜爱的 Shell（Bash、Powershell 或 Z shell）来对 Amazon Web Services 服务 运行 Amazon CLI 命令。在执行此操作时，您无需下载或安装命令行工具。有关更多信息，请参阅《Amazon CloudShell 用户指南》中的什么是 Amazon CloudShell？。
- 不要为需要访问应用程序或 Amazon Web Services 服务 的人类用户创建长期访问密钥 – IAM Identity Center 可以生成临时访问凭证，供您的外部 IdP 用户访问 Amazon Web Services 服务。这样就无需在 IAM 中创建和管理长期凭证。在 IAM Identity Center 中，创建一个 IAM Identity Center 权限集，该权限集向外部 IdP 用户授予访问权限。然后，将来自 IAM Identity Center 的组分配给选定 Amazon Web Services 账户 中的权限集。有关更多信息，请参阅什么是 Amazon IAM Identity Center、连接到外部身份提供者以及 Amazon IAM Identity Center 用户指南中的权限集。
- 不要 Amazon 计算服务中存储长期访问密钥 – 相反，应为计算资源分配 IAM 角色。这会自动提供临时凭证以授予访问权限。例如，在创建附加到 Amazon EC2 实例的实例配置文件，可以将 Amazon 角色分配给该实例并使其对该实例的所有应用程序可用。实例配置文件包含角色，并使 Amazon EC2 实例上运行的程序能够获得临时凭证。要了解更多信息，请参阅使用 IAM 角色为 Amazon EC2 实例上运行的应用程序授予权限。

Amazon 安全审核指南

定期审查安全配置，以确保其满足您当前的业务需求。审查可以为您提供机会，删除不需要的 IAM 用户、角色、组和策略，以确保您的用户和软件不会拥有过多的权限。

以下是有关系统地查看和监控 Amazon 资源的准则，以便获得安全最佳实践。

> **Tip**
>
> 您可以监控 IAM 的使用情况，因为它与使用 Amazon Security Hub CSPM 的安全最佳实践有关。Security Hub CSPM 使用安全控件来评估资源配置和安全标准，以帮助您遵守各种合规框架。有关使用 Security Hub CSPM 评估 IAM 资源的更多信息，请参阅《Amazon Security Hub CSPM User Guide》中的 Amazon Identity and Access Management controls。

内容

- 何时执行安全审查
- 审核准则
- 审核 Amazon 账户凭证
- 审核 IAM 用户
- 审核 IAM 组
- 审核 IAM 角色
- 查看您的 SAML 和 OpenID Connect（OIDC）的 IAM 提供商
- 审核移动应用程序
- 有关审核 IAM policy 的提示

何时执行安全审查

遇到以下情况时，您应该审查您的安全配置：

- 定期。将本文档中介绍的步骤作为安全最佳实践并定期执行。
- 如果组织中发生变动，比如有人离职。
- 您已停止使用一项或多项单独 Amazon 服务，以验证您已删除账户中的用户不再需要的权限。
- 如果您在账户中添加或删除了软件，比如 Amazon EC2 实例、Amazon OpsWorks 堆栈、Amazon CloudFormation 模板等内容上的应用程序。

- 您怀疑某个未授权人员可能访问了您的账户。

审核准则

查看您账户的安全配置时，请遵循这些准则：

- 全面周祥。查看安全配置的各个方面，包括您很少使用的那些方面。
- 请勿假设。如果您对安全配置的某些方面（例如，特定策略背后的理由或角色存在情况）不太熟悉，请调查业务需求，直到您了解潜在风险。
- 让事情变得简单。为了使审查（和管理）变得更简单，请使用 IAM 组、IAM 角色、一致的命名方案和简单的策略。

审核 Amazon 账户凭证

当审核您的 Amazon 账户证书时，执行以下步骤：

1. 如果您有根用户的访问密钥，但您没有使用，则可以将其删除。我们强烈建议您使用拥有临时凭证的用户,例如 Amazon IAM Identity Center 中的用户，而不是使用根访问密钥来完成 Amazon 日常工作。
2. 如果您的账户需要访问密钥，请务必 在需要时进行更新。

审核 IAM 用户

当您审计您的现有 IAM 用户时，请执行以下步骤：

1. 列出您的用户，然后删除不需要的用户。
2. 将用户从其无需访问的群组中删除。
3. 查看附加到用户所在的组的策略。请参阅有关审核 IAM policy 的提示。
4. 删除用户不需要或者可能已经公开的安全证书。例如，应用程序的 IAM 用户无需密码（只有登录 Amazon 网站才需要密码）。同样，如果用户不使用访问密钥，则不必拥有访问密钥。有关更多信息，请参阅管理 IAM 用户的密码和管理 IAM 用户的访问密钥。

   您可以生成和下载列出您账户中所有 IAM 用户及其各个凭证状态（包括密码、访问密钥和 MFA 设备）的凭证报告。对于密码和访问密钥，凭证报告将显示密码或访问密钥最近一次的使用日期和时间。请考虑从您的账户中删除最近未使用的凭证。（切勿删除您的紧急访问用户。）有关更多信息，请参阅获取您 Amazon 账户的凭证报告。

5. 对于需要长期凭证的用例，应在需要时更新密码和访问密钥。有关更多信息，请参阅管理 IAM 用户的密码和管理 IAM 用户的访问密钥。
6. 作为最佳实践，请要求人类用户必须使用带有身份提供者的联合身份验证才能使用临时凭证访问 Amazon。如果可以，请从 IAM 用户过渡到联合主体，例如 IAM Identity Center 的用户。保留应用程序所需的最低 IAM 用户数。

审核 IAM 组

当您审计您的 IAM 组时，请执行以下步骤：

1. 列出您的组，然后删除未使用的组。
2. 查看用户（位于每个组中）并删除用户（不属于这些组）。
3. 查看附加到组的策略。请参阅有关审核 IAM policy 的提示。

审核 IAM 角色

当您审计您的 IAM 角色时，请执行以下步骤：

1. 列出您的角色，然后删除未使用的角色。
2. 查看角色的信任策略。确保您知道委托人是谁，并且了解为什么账户或用户需要能够担任该角色。
3. 查看角色的访问策略，以确保其向担任该角色的人授予了合适的权限，请参阅有关审核 IAM policy 的提示。

查看您的 SAML 和 OpenID Connect（OIDC）的 IAM 提供商

如果您已创建 IAM 实体来与 SAML 或 OIDC 身份提供者（IdP）建立信任关系，请执行以下步骤：

1. 删除未使用的提供商。
2. 下载并查看每个 SAML IdP 的 Amazon 元数据文档，并确保这些文档反映了您当前的业务需求。
3. 从 SAML IdP 获取最新元数据文档，并在 IAM 中更新提供者。

审核移动应用程序

如果您已经创建了向 Amazon 提出请求的移动应用程序，请执行以下步骤：

1. 确保移动应用程序不包含嵌入式访问密钥（即使它们位于加密存储中）。
2. 通过使用出于此目的而设计的 API 来获取应用程序的临时凭证。

> **Note**
>
> 我们建议您使用 Amazon Cognito 来管理应用程序中的用户身份。此服务让您可以使用 Login with Amazon、Facebook、Google 或任何与 OpenID Connect（OIDC）兼容的身份提供商进行用户身份验证。有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的 Amazon Cognito 用户池。

有关审核 IAM policy 的提示

策略功能强大且非常细微，因此，学习并了解每个策略授予的权限很重要。查看策略时请使用以下准则：

- 将策略附加到组或角色而不是单个用户。如果单个用户拥有策略，请确保您了解为什么该用户需要策略。
- 确保 IAM 用户、组及角色拥有所需的权限，并且没有任何额外的权限。
- 使用 IAM Policy Simulator 对附加到用户或组的策略进行测试。
- 请记住，用户的权限是所有适用策略的结果，适用策略包括基于身份的策略（如用户策略、组策略或角色策略）和基于资源的策略（Amazon S3 存储桶、Amazon SQS 队列、Amazon SNS 主题和 Amazon KMS 密钥等资源的策略）。检查应用于用户的所有策略以及了解授予单个用户的一整套权限很重要。
- 请注意，通过允许用户创建 IAM 用户、组、角色或策略，并将策略附加到主要实体，可以有效地向用户授予针对账户中所有资源的权限。可创建策略并将其附加到用户、组或角色的用户能为自己授予任何权限。一般而言，请勿向您不信任的用户或角色授予可以完全访问账户中资源的 IAM 权限。进行安全审查时，请确认已向可信身份授予以下 IAM 权限：
  - `iam:PutGroupPolicy`
  - `iam:PutRolePolicy`
  - `iam:PutUserPolicy`
  - `iam:CreatePolicy`
  - `iam:CreatePolicyVersion`
  - `iam:AttachGroupPolicy`

  - `iam:AttachRolePolicy`
  - `iam:AttachUserPolicy`
- 确保策略没有向您未使用的服务授予权限。例如，如果使用 Amazon 托管策略，请确保您账户中正在使用的 Amazon 托管策略是针对您实际使用的服务的。要找出您账户中正在使用哪些 Amazon 托管式策略，请使用 IAM `GetAccountAuthorizationDetails` API（Amazon CLI 命令：`aws iam get-account-authorization-details`）。
- 如果策略授予用户启动 Amazon EC2 实例的权限，这也可能允许 `iam:PassRole` 操作，但如果是这样，其应该明确列出用户可以传递给 Amazon EC2 实例的角色。
- 请检查包括 `*` 的 `Action` 或 `Resource` 元素的任何值。如果可以，授予用户所需的个人操作和资源的 `Allow` 访问权限。但是，以下是可能适合在策略中使用 `*` 的原因：
  - 该策略旨在授予管理级权限。
  - 为方便起见，通配符用于一组相似的操作（例如，`Describe*`），您会因为以这种方式引用的操作的完整列表而感到轻松。
  - 通配符用于表示一类资源或一个资源路径（例如，`arn:aws:iam::`*`account-id`*`:users/division_abc/*`），您可以很方便地授予针对该类别或路径中所有资源的访问权限。
  - 服务操作不支持资源级权限，资源的唯一选择是 `*`。
- 检查策略名称以确保其反映了策略的功能。例如，尽管策略名称可能包括“只读”，但策略可能实际还授予了写入或更改权限。

有关规划安全审计的更多信息，请参阅《Amazon 架构中心》中的安全、身份和合规性方面的最佳实践。

Amazon Identity and Access Management 中的数据保护

Amazon 责任共担模式适用于保护 Amazon Identity and Access Management 中的数据。如该模式中所述，Amazon 负责保护运行所有 Amazon Web Services 云 的全球基础结构。您负责维护对托管在此基础结构上的内容的控制。您还负责您所使用的 Amazon Web Services 服务 的安全配置和管理任务。有关数据隐私的更多信息，请参阅Amazon 中国区域法律条款。

出于数据保护目的，建议您保护 Amazon Web Services 账户 凭证并使用 Amazon IAM Identity Center 或 Amazon Identity and Access Management（IAM）设置单个用户。这样，每个用户只获得履行其工作职责所需的权限。还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与 Amazon 资源进行通信。我们要求使用 TLS 1.2，建议使用 TLS 1.3。

- 使用 Amazon CloudTrail 设置 API 和用户活动日记账记录。有关使用 CloudTrail 跟踪来捕获 Amazon 活动的信息，请参阅《Amazon CloudTrail 用户指南》中的使用 CloudTrail 跟踪。
- 使用 Amazon 加密解决方案以及 Amazon Web Services 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的敏感数据。
- 如果在通过命令行界面或 API 访问 Amazon 时需要经过 FIPS 140-3 验证的加密模块，请使用 FIPS 端点。有关可用的 FIPS 端点的更多信息，请参阅《美国联邦信息处理标准（FIPS）第 140-3 版》https://www.amazonaws.cn/compliance/fips/。

强烈建议您切勿将机密信息或敏感信息（如您客户的电子邮件地址）放入标签或自由格式文本字段（如名称字段）。这包括处理 IAM 或其他 Amazon Web Services 服务 时使用控制台、API、Amazon CLI 或 Amazon 开发工具包。在用于名称的标签或自由格式文本字段中输入的任何数据都可能会用于计费或诊断日志。如果您向外部服务器提供 URL，强烈建议您不要在网址中包含凭证信息来验证对该服务器的请求。

IAM 和 Amazon STS 中的数据加密

数据加密通常分为两类：静态加密和传输中加密。

静态加密

将对由 IAM 收集和存储的数据进行静态加密。

- IAM – 在 IAM 中收集和存储的数据包括 IP 地址、客户账户元数据和客户识别数据（包括密码）。客户账户元数据和客户识别数据使用 AES 256 进行静态加密或使用 SHA 256 进行哈希处理。
- Amazon STS - Amazon STS 不收集客户内容（用于记录针对服务的成功请求和有错误的请求的服务日志除外）。

传输中加密

客户身份识别数据（包括密码）将使用 TLS 1.2 和 1.3 进行传输中加密。所有 Amazon STS 终端节点都支持 HTTPS 来对传输中的数据进行加密。有关 Amazon STS 终端节点的列表，请参阅Amazon STS 区域和端点。

IAM 和 Amazon STS 中的密钥管理

您无法使用 IAM 或 Amazon STS 管理加密密钥。有关加密密钥的更多信息，请参阅 Amazon Key Management Service 开发人员指南中的什么是 Amazon KMS？。

IAM 和 Amazon STS 中的互联网络流量保密性

必须使用传输层安全协议 (TLS) 发出对 IAM 的请求。可以使用 VPC 终端节点保护与 Amazon STS 服务的连接。要了解更多信息，请参阅接口 VPC 端点。

Amazon Identity and Access Management 中的日志记录和监控

监控是使您的 Amazon Identity and Access Management (IAM)、Amazon Security Token Service (Amazon STS) 和您的其他 Amazon 解决方案保持可靠性、可用性和性能的重要环节。Amazon 提供了多种工具，用于监控您 Amazon 资源和对潜在的事件作出响应：

- Amazon CloudTrail 将对 IAM 和 Amazon STS 的所有 API 调用作为事件捕获，包括来自控制台的调用和 API 调用。要了解有关将 CloudTrail 与 IAM 和 Amazon STS 搭配使用的更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》https://docs.amazonaws.cn/awscloudtrail/latest/userguide/。
- Amazon Identity and Access Management and Access Analyzer 帮助您识别您的组织和账户中与外部实体共享的资源（例如 Amazon S3 存储桶或 IAM 角色）。这可以帮助您识别对资源和数据的意外访问，此类访问会带来安全风险。要了解更多信息，请参阅 什么是 IAM 访问分析器？
- Amazon CloudWatch 实时监控您的 Amazon 资源以及在 Amazon 上运行的应用程序。您可以收集和跟踪指标，创建自定义的控制平面，以及设置警报以在指定的指标达到您指定的阈值时通知您或采取措施。例如，您可以使用 CloudWatch 跟踪 Amazon EC2 实例的 CPU 使用率或其他指标并且在需要时自动启动新实例。有关更多信息，请参阅《Amazon CloudWatch 用户指南》。
- Amazon CloudWatch Logs 帮助您监控、存储和访问来自 Amazon EC2 实例、CloudTrail 和其他来源的日志文件。CloudWatch Logs 可以监控日志文件中的信息，并在达到特定阈值时通知您。您还可以在高持久性存储中检索您的日志数据。有关更多信息，请参阅 Amazon CloudWatch Logs 用户指南。

有关 IAM 的其他资源和安全最佳实践，请参阅 Amazon Identity and Access Management 中的安全最佳实践和使用案例。

主题

- 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用
- 在 Amazon CloudTrail 中跟踪特权任务

使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用

IAM 和 Amazon STS 与 Amazon CloudTrail 集成，后者是记录 IAM 用户或角色所执行操作的服务。CloudTrail 将对 IAM 和 Amazon STS 的所有 API 调用作为事件捕获，包括来自控制台的调用和 API 调用。如果您创建了跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。可使用 CloudTrail 获取有关对 IAM 或 Amazon STS 发出的请求的信息。例如，您可以查看发出请求的源 IP 地址、用户、时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》。

主题

- CloudTrail 中的 IAM 和 Amazon STS 信息
- 记录 IAM 和 Amazon STS API 请求
- 将 API 请求记录到其他 Amazon 服务
- 记录用户登录事件
- 记录临时凭证的登录事件
- CloudTrail 日志中的 IAM API 事件示例
- CloudTrail 日志中的 Amazon STS API 事件示例
- CloudTrail 日志中的登录事件示例
- IAM 角色信任策略行为

CloudTrail 中的 IAM 和 Amazon STS 信息

在您创建 Amazon Web Services 账户时，将在该账户上启用 CloudTrail。当 IAM 或 Amazon STS 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在 Event history（事件历史记录）中。您可以在 Amazon Web Services 账户 中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 Amazon Web Services 账户 中的事件（包括 IAM 和 Amazon STS 的事件），请创建跟踪。通过跟踪，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅：

- 创建跟踪概述

- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

所有 IAM 和 Amazon STS 操作均由 CloudTrail 记录，并记录在了 IAM API 参考和 Amazon Security Token Service API 参考中。

记录 IAM 和 Amazon STS API 请求

CloudTrail 将所有经过身份验证的 API 请求记录到 IAM 和 Amazon STS API 操作。CloudTrail 还会将未经身份验证的请求记录到 Amazon STS 操作、`AssumeRoleWithSAML` 和 `AssumeRoleWithWebIdentity`，并记录身份提供程序所提供的信息。但是，某些未经身份验证的 Amazon STS 请求可能不会被记录，因为它们不符合最低期望，即其有效性不足以作为合法请求被信任。对于跨账户角色假设和通过 `AssumeRoot` API 启动的特权会话，CloudTrail 不会在目标账户中记录被拒绝的 Amazon STS 请求。

您可以使用记录的信息将由一个 OIDC 或 SAML 联合主体通过担任角色发出的调用映射回原始外部联合调用方。如果使用 `AssumeRole`，您可以将调用映射回到发出请求的 Amazon; 服务或发出请求的用户的账户。CloudTrail 日志条目中 JSON 数据的 `userIdentity` 部分包含您将 AssumeRole* 请求与特定会话主体进行映射时所需的信息。有关更多信息，请参阅《Amazon CloudTrail 用户指南》中的 CloudTrail userIdentity 元素。

IAM 用户使用 MFA 登录时，Amazon CloudTrail 日志将包含 MFA 信息。如果 IAM 用户代入某个 IAM 角色，CloudTrail 还将在利用代入角色执行操作的 `sessionContext` 属性中记录 `mfaAuthenticated: true`。但是，当使用代入角色的凭证进行 API 调用时，CloudTrail 日志记录与 IAM 要求的内容是分开的。有关更多信息，请参阅 CloudTrail userIdentity 元素。

例如，IAM `CreateUser`、`DeleteRole`、`ListGroups` 调用和其他 API 操作全部都由 CloudTrail 记录。

本主题后面将会介绍有关此类日志条目的示例。

将 API 请求记录到其他 Amazon 服务

对其他 Amazon 服务 API 操作的经过身份验证的请求由 CloudTrail 记录，这些日志条目包含有关生成请求的人员的信息。

例如，假设您发出一个请求来列出 Amazon EC2 实例或创建 Amazon CodeDeploy 部署组。有关发出该请求的人员或服务的详细信息将包含在该请求的日志条目中。此信息可帮助您确定发出该请求的是 Amazon Web Services 账户根用户、IAM 用户、角色还是其他 Amazon 服务。

有关 CloudTrail 日志条目中用户身份信息的详细信息，请参阅 Amazon CloudTrail 用户指南中的 userIdentity 元素。

记录用户登录事件

CloudTrail 会将登录事件记录到 Amazon Web Services 管理控制台、Amazon CLI 和 SDK 等本地开发工具、Amazon 论坛以及 Amazon Web Services Marketplace。CloudTrail 将为 IAM 用户、SAML 和 OIDC 联合主体，以及 Amazon STS 联合用户主体记录成功和失败的登录尝试。

要查看成功和不成功的根用户登录的 CloudTrail 事件示例，请参阅 Amazon CloudTrail 用户指南中的根用户的事件记录示例。

作为最佳安全做法，当用户名称不正确导致登录故障时，Amazon 不记录输入的 IAM 用户名称文本。用户名称文本将用 HIDDEN_DUE_TO_SECURITY_REASONS 值代替。有关示例，请参阅本主题后面的用户名称不正确导致的登录失败事件示例。用户名称文本隐藏，因为此类错误可能是用户错误所导致。记录这些错误可能会泄露潜在的敏感信息。例如：

- 您不小心在用户名框中键入了密码。
- 您选择一个 Amazon Web Services 账户 的登录页面的链接，但随后键入了另一 Amazon Web Services 账户 的账号。
- 您忘记了所登录的是哪个账户，不小心键入了个人电子邮件账户的账户名、银行登录标识符或某些其他私有 ID。

记录临时凭证的登录事件

当主体请求临时凭证时，主体类型将决定 CloudTrail 记录事件的方式。当主体代入其他账户中的角色时，这会很复杂。有多个 API 调用执行与角色跨账户操作相关的操作。首先，主体调用 Amazon STS API 来检索临时凭证。该操作记录到调用账户中以及执行 Amazon STS 操作的账户中。然后，主体使用角色在代入角色的账户中执行其他 API 调用。

您可以在角色信任策略中使用 `sts:SourceIdentity` 条件键，以要求用户在代入角色时指定身份。例如，您可以要求 IAM 用户指定自己的用户名作为其源身份。这可以帮助您确定哪个用户在 Amazon 中执行了具体的操作。有关更多信息，请参阅 `sts:SourceIdentity`。您可以使用 `sts:RoleSessionName`，以要求用户在代入角色时指定会话名称。这可以在您查看 Amazon CloudTrail 日志时帮助您区分不同主体对角色使用的角色会话。

下表说明了 CloudTrail 如何为生成临时凭证的每个 Amazon STS API 记录不同的用户标识信息。

主体类型	STS API	发起人账户的 CloudTrail 日志中的用户身份	所代入角色的账户的 CloudTrail 日志中的用户身份	角色后续 API 调用的 CloudTrail 日志中的用户身份
Amazon Web Services 账户根用户 凭证	GetSessionToken	根用户身份	角色所有者账户与调用账户相同	根用户身份
Amazon Web Services 账户根用户 凭证	AssumeRoot	根用户会话	账号和主体 ID（如果是用户）	根用户会话
IAM 用户	GetSessionToken	IAM 用户身份。	角色所有者账户与调用账户相同	IAM 用户身份。
IAM 用户	GetFederationToken	IAM 用户身份。	角色所有者账户与调用账户相同	IAM 用户身份。
IAM 用户	AssumeRole	IAM 用户身份。	账号和主体 ID（如果是用户）或 Amazon 服务主体	仅角色身份（无用户）
外部验证的用户	AssumeRoleWithSAML	不适用	SAML 用户身份	仅角色身份（无用户）
外部验证的用户	AssumeRoleWithWebIdentity	不适用	OIDC/Web 用户身份	仅角色身份（无用户）

如果某项操作对资源没有任何变异影响，那么 CloudTrail 会将其视为只读。当记录只读事件时，CloudTrail 会编辑日志中的 `responseElements` 信息。当 CloudTrail 记录非只读事件时，日志条目中会显示完整的 `responseElements`。对于 Amazon STS API `AssumeRole`、`AssumeRoleWithSAML` 和 `AssumeRoleWithWebIdentity`，即使它们被记录为只读，CloudTrail 也会在这些 API 的日志中包含完整的 `responseElements`（`secretAccessKey` 除外）。

下表说明了 CloudTrail 如何为生成临时凭证的每个 Amazon STS API 记录 `responseElements` 和 `readOnly` 信息。

STS API	响应元素信息	Read-only
AssumeRole	包含	true
AssumeRoleWithSAML	包含	true
AssumeRoleWithWebIdentity	包含	true
AssumeRoot	包含	false
GetFederationToken	包含	false
GetSessionToken	包含	false

CloudTrail 日志中的 IAM API 事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。一个 API 事件表示一个 API 请求，并包括有关主体、所请求的操作、任意参数以及该操作日期和时间的信息。

CloudTrail 日志文件中的 IAM API 事件示例

以下示例说明了为进行 IAM `GetUserPolicy` 操作而发出的请求的 CloudTrail 日志条目。

```
{
    "eventVersion": "1.09",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:Role-Session-Name",
        "arn": "arn:aws:sts::111122223333:assumed-role/Role-Name/Role-Session-Name",
        "accountId": "111122223333",
        "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
```

```
            },
            "attributes": {
                "creationDate": "2024-09-09T17:50:16Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-09-09T17:51:44Z",
    "eventSource": "iam.amazonaws.com",
    "eventName": "GetUserPolicy",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.101",
    "userAgent": "aws-cli/1.16.96 Python/2.7.8 Linux/10 botocore/1.12.86",
    "requestParameters": {
        "userName": "ExampleIAMUserName",
        "policyName": "ExamplePoliccyName"
    },
    "responseElements": null,
    "requestID": "9EXAMPLE-0c68-11e4-a24e-d5e16EXAMPLE",
    "eventID": "cEXAMPLE-127e-4632-980d-505a4EXAMPLE",
    "readOnly": true,
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "111122223333",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "iam.amazonaws.com"
    }
}
```


通过此事件信息，您可以确定发出该请求的目的是为了获取用户 ReadOnlyAccess-JaneDoe-201407151307 的名为 JaneDoe 的用户策略，如 requestParameters 元素中所述。您还可以看到，该请求是由名为 JaneDoe 的 IAM 用户在 2014 年 7 月 15 日下午 9:40 (UTC) 发出的。在本示例中，请求源自 Amazon Web Services 管理控制台，您可以从 userAgent 元素中判断出来。

CloudTrail 日志中的 Amazon STS API 事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。一个 API 事件表示一个 API 请求，并包括有关主体、所请求的操作、任意参数以及该操作日期和时间的信息。

CloudTrail 日志文件中跨账户 Amazon STS API 事件的示例

账户 777788889999 中名为 John 的 IAM 用户调用 Amazon STS AssumeRole 操作来在账户 111122223333 中担任角色 EC2-dev。账户管理员要求用户在担任角色时设置与其用户名相同的源身份。用户传入 John 的源身份值。


```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "arn": "arn:aws:iam::777788889999:user/John",
    "accountId": "777788889999",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "userName": "John"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "John-EC2-dev",
    "sourceIdentity": "John",
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2023, 4:07:39 PM"
      },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:John-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/John-EC2-dev"
    },
  "sourceIdentity": "John"
  },
  "resources": [
    {
      "ARN": "arn:aws:iam::111122223333:role/EC2-dev",
```

```
      "accountId": "111122223333",
      "type": "AWS::IAM::Role"
    }
  ],
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "111122223333"
}
```

第二个示例显示同一请求的代入角色账户 (111122223333) 的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "AWSAccount",
    "principalId": "AIDAQRSTUVWXYZEXAMPLE",
    "accountId": "777788889999"
  },
  "eventTime": "2014-07-18T15:07:39Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRole",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67",
  "requestParameters": {
    "roleArn": "arn:aws:iam::111122223333:role/EC2-dev",
    "roleSessionName": "John-EC2-dev",
    "sourceIdentity": "John",
  },
  "responseElements": {
    "credentials": {
      "sessionToken": "<encoded session token blob>",
      "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
      "expiration": "Jul 18, 2014, 4:07:39 PM"
    },
    "assumedRoleUser": {
      "assumedRoleId": "AIDAQRSTUVWXYZEXAMPLE:John-EC2-dev",
      "arn": "arn:aws:sts::111122223333:assumed-role/EC2-dev/John-EC2-dev"
      },
  "sourceIdentity": "John"
```

```
  },
  "requestID": "4EXAMPLE-0e8d-11e4-96e4-e55c0EXAMPLE",
  "sharedEventID": "bEXAMPLE-efea-4a70-b951-19a88EXAMPLE",
  "eventID": "dEXAMPLE-ac7f-466c-a608-4ac8dEXAMPLE"
}
```

CloudTrail 日志文件中 Amazon STS 角色链接 API 事件示例

以下示例演示账户 111111111111 中 John Doe 发出的请求的 CloudTrail 日志条目。John 之前使用自己的 John 用户代入 JohnRole1 角色。对于此请求，他使用来自该角色的凭证代入 JohnRole2 角色。这称为角色链。当他担任 John1 角色时设置的源身份将在担任 JohnRole2 的请求中继续存在。如果 John 尝试在担任角色时设置不同的源身份，请求将被拒绝。John 将两个会话标签传递到请求中。他将这两个标签设置为可传递。请求继承 Department 标签作为可传递标签，因为 John 在代入 JohnRole1 时将其设置为可传递。更多有关源身份的信息，请参阅 监控和控制使用所担任角色执行的操作。有关角色链中可传递键的更多信息，请参阅使用会话标签链接角色。

```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROAIN5ATK5U7KEXAMPLE:JohnRole1",
        "arn": "arn:aws:sts::111111111111:assumed-role/John/JohnRole1",
        "accountId": "111111111111",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2019-10-02T21:50:54Z"
            },
                "sessionIssuer": {
                "type": "Role",
               "principalId": "AROAIN5ATK5U7KEXAMPLE",
                "arn": "arn:aws:iam::111111111111:role/JohnRole1",
                "accountId": "111111111111",
                "userName": "John"
            },
            "sourceIdentity": "John"
        }
    },
    "eventTime": "2019-10-02T22:12:29Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
```

```
    "awsRegion": "us-east-2",
    "sourceIPAddress": "123.145.67.89",
    "userAgent": "aws-cli/1.16.248 Python/3.4.7
 Linux/4.9.184-0.1.ac.235.83.329.metal1.x86_64 botocore/1.12.239",
    "requestParameters": {
        "incomingTransitiveTags": {
            "Department": "Engineering"
        },
        "tags": [
            {
                "value": "johndoe@example.com",
                "key": "Email"
            },
            {
                "value": "12345",
                "key": "CostCenter"
            }
        ],
        "roleArn": "arn:aws:iam::111111111111:role/JohnRole2",
        "roleSessionName": "Role2WithTags",
        "sourceIdentity": "John",
        "transitiveTagKeys": [
            "Email",
            "CostCenter"
        ],
        "durationSeconds": 3600
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
            "expiration": "Oct 2, 2019, 11:12:29 PM",
            "sessionToken": "AgoJb3JpZ2luX2VjEB4aCXVzLXdlc3QtMSJHMEXAMPLETOKEN
+//rJb8Lo30mFc5MlhFCEbubZvEj0wHB/mDMwIgSEe9gk/Zjr09tZV7F1HDTMhmEXAMPLETOKEN/iEJ/
rkqngII9//////////
ARABGgw0MjgzMDc4NjM5NjYiDLZjZFKwP4qxQG5sFCryASO4UPz5qE97wPPH1eLMvs7CgSDBSWfonmRTCfokm2FN1+hWUdQ
+C+WKFZb701eiv9J5La2EXAMPLETOKEN/c7S5Iro1WUJ0q3Cxuo/8HUoSxVhQHM7zF7mWWLhXLEQ52ivL
+F6q5dpXu4aTFedpMfnJa8JtkWwG9x1Axj0Ypy2ok8v5unpQGWych1vwdvj6ez1Dm8Xg1+qIzXILiEXAMPLETOKEN/
vQGqu8H+nxp3kabcrtOvTFTvxX6vsc8OGwUfHhzAfYGEXAMPLETOKEN/
L6v1yMM3B1OwFOrQBno1HEjf1oNI8RnQiMNFdUOtwYj7HUZIOCZmjfN8PPHq77N7GJl9lzvIZKQA0Owcjg
+mc78zHCj8y0siY8C96paEXAMPLETOKEN/
E3cpksxWdgs91HRzJWScjN2+r2LTGjYhyPqcmFzzo2mCE7mBNEXAMPLETOKEN/oJy
+2o83YNW5tOiDmczgDzJZ4UKR84yGYOMfSnF4XcEJrDgAJ3OJFwmTcTQICAlSwLEXAMPLETOKEN"
        },
        "assumedRoleUser": {
```

```
            "assumedRoleId": "AROAIFR7WHDTSOYQYHFUE:Role2WithTags",
            "arn": "arn:aws:sts::111111111111:assumed-role/test-role/Role2WithTags"
        },
    "sourceIdentity": "John"
    },
    "requestID": "b96b0e4e-e561-11e9-8b3f-7b396EXAMPLE",
    "eventID": "1917948f-3042-46ec-98e2-62865EXAMPLE",
    "resources": [
        {
            "ARN": "arn:aws:iam::111111111111:role/JohnRole2",
            "accountId": "111111111111",
            "type": "AWS::IAM::Role"
        }
    ],
    "eventType": "AwsApiCall",
    "recipientAccountId": "111111111111"
}
```

CloudTrail 日志文件中 Amazon 服务 Amazon STS API 事件的示例

以下示例显示由 Amazon 服务使用服务角色中的权限调用其他服务 API 发出的请求的 CloudTrail 日志条目。它显示了在 777788889999 账户中发出的请求的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.04",
  "userIdentity": {
    "type": "AssumedRole",
    "principalId": "AROAQRSTUVWXYZEXAMPLE:devdsk",
    "arn": "arn:aws:sts::777788889999:assumed-role/AssumeNothing/devdsk",
    "accountId": "777788889999",
    "accessKeyId": "ASIAI44QH8DHBEXAMPLE",
    "sessionContext": {
      "attributes": {
        "creationDate": "2016-11-14T17:25:26Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AROAQRSTUVWXYZEXAMPLE",
        "arn": "arn:aws:iam::777788889999:role/AssumeNothing",
        "accountId": "777788889999",
        "userName": "AssumeNothing"
      }
    }
```

```
  },
  "eventTime": "2016-11-14T17:25:45Z",
  "eventSource": "s3.amazonaws.com",
  "eventName": "DeleteBucket",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.1",
  "userAgent": "[aws-cli/1.11.10 Python/2.7.8
 Linux/3.2.45-0.6.wd.865.49.315.metal1.x86_64 botocore/1.4.67]",
  "requestParameters": {
    "bucketName": "amzn-s3-demo-bucket"
  },
  "responseElements": null,
  "requestID": "EXAMPLE463D56D4C",
  "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
  "eventType": "AwsApiCall",
  "recipientAccountId": "777788889999"
}
```

CloudTrail 日志文件中 SAML Amazon STS API 事件的示例

以下示例说明了为进行 Amazon STS AssumeRoleWithSAML 操作而发出的请求的 CloudTrail 日志条目。请求包括 SAML 属性 `CostCenter` 和 `Project`，这些属性作为会话标签通过 SAML 断言传递。这些标签被设置为可传递标签，因此在角色链场景中持续存在。该请求包含可选的 API 参数 `DurationSeconds`，在 CloudTrail 日志中表示为 `durationSeconds`，并设置为 1800 秒。该请求还包含在 SAML 断言中传递的 SAML 属性 `sourceIdentity`。如果某人使用生成的角色会话凭证担任另一个角色，则此源身份将仍然存在。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "SAMLUser",
        "principalId": "SampleUkh1i4+ExamplexL/jEvs=:SamlExample",
        "userName": "SamlExample",
        "identityProvider": "bdGOnTesti4+ExamplexL/jEvs="
    },
    "eventTime": "2023-08-28T18:30:58Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoleWithSAML",
    "awsRegion": "us-east-2",
    "sourceIPAddress": "AWS Internal",
    "userAgent": "aws-internal/3 aws-sdk-java/1.12.479
 Linux/5.10.186-157.751.amzn2int.x86_64 OpenJDK_64-Bit_Server_VM/17.0.7+11 java/17.0.7
 kotlin/1.3.72 vendor/Amazon.com_Inc. cfg/retry-mode/standard",
```

```
    "requestParameters": {
        "sAMLAssertionID": "_c0046cEXAMPLEb9d4b8eEXAMPLE2619aEXAMPLE",
        "roleSessionName": "MyAssignedRoleSessionName",
        "sourceIdentity": "MySAMLUser",
        "principalTags": {
            "CostCenter": "987654",
            "Project": "Unicorn",
            "Department": "Engineering"
        },
        "transitiveTagKeys": [
            "CostCenter",
            "Project"
        ],
        "roleArn": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth",
        "principalArn": "arn:aws:iam::444455556666:saml-provider/Shibboleth",
        "durationSeconds": 1800
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "<encoded session token blob>",
            "expiration": "Aug 28, 2023, 7:00:58 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROAD35QRSTUVWEXAMPLE:MyAssignedRoleSessionName",
            "arn": "arn:aws:sts::444455556666:assumed-role/SAMLTestRoleShibboleth/
MyAssignedRoleSessionName"
        },
        "packedPolicySize": 1,
        "subject": "SamlExample",
        "subjectType": "transient",
        "issuer": "https://server.example.com/idp/shibboleth",
        "audience": "https://signin.aws.amazon.com/saml",
        "nameQualifier": "bdGOnTesti4+ExamplexL/jEvs=",
        "sourceIdentity": "MySAMLUser"
    },
    "requestID": "6EXAMPLE-e595-11e5-b2c7-c974fEXAMPLE",
    "eventID": "dEXAMPLE-265a-41e0-9352-4401bEXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::Role",
            "ARN": "arn:aws:iam::444455556666:role/SAMLTestRoleShibboleth"
```

```
        },
        {
            "accountId": "444455556666",
            "type": "AWS::IAM::SAMLProvider",
            "ARN": "arn:aws:iam::444455556666:saml-provider/test-saml-provider"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "444455556666",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.2",
        "cipherSuite": "ECDHE-RSA-AES128-GCM-SHA256",
        "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
    }
}
```

CloudTrail 日志文件中 OIDC Amazon STS API 事件示例

以下示例说明了为进行 Amazon STS AssumeRoleWithWebIdentity 操作而发出的请求的 CloudTrail 日志条目。请求包括属性 CostCenter 和 Project，这些属性作为会话标签通过 OpenID Connect（OIDC）身份提供商（IdP）令牌传递。这些标签被设置为可传递标签，因此在角色链中持续存在。请求包含身份提供程序令牌中的 sourceIdentity 属性。如果某人使用生成的角色会话凭证担任另一个角色，则此源身份将仍然存在。

CloudTrail 日志条目还包含一个带有 identityProviderConnectionVerificationMethod 属性的 additionalEventData 字段。此属性表示 Amazon 用于验证与 OIDC 提供商的连接的方法。属性值将为 IAMTrustStore 或 Thumbprint。IAMTrustStore 值表示 Amazon 使用我们的可信根凭证颁发机构（CA）库成功验证了与 OIDC IdP 的连接。Thumbprint 值表示 Amazon 使用在 IdP 配置中设置的证书指纹来验证 OIDC IdP 服务器证书。

```
{
  "eventVersion": "1.08",
  "userIdentity": {
    "type": "WebIdentityUser",
    "principalId": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>:<id of application>:<id of user>",
    "userName": "<id of user>",
    "identityProvider": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>"
  },
```

```
  "eventTime": "2024-07-09T15:41:37Z",
  "eventSource": "sts.amazonaws.com",
  "eventName": "AssumeRoleWithWebIdentity",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "aws-cli/2.13.29 Python/3.11.6 Windows/10 exe/AMD64 prompt/off command/
sts.assume-role-with-web-identity",
  "requestParameters": {
    "roleArn": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole",
    "roleSessionName": "<assigned role session name>",
    "sourceIdentity": "MyWebIdentityUser",
    "durationSeconds": 3600,
    "principalTags": {
      "CostCenter": "24680",
      "Project": "Pegasus"
    },
    "transitiveTagKeys": [
      "CostCenter",
      "Project"
    ]
  },
  "responseElements": {
    "credentials": {
      "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
      "sessionToken": "<encoded session token blob>",
      "expiration": "Jul 9, 2024, 4:41:37 PM"
    },
    "subjectFromWebIdentityToken": "<id of user>",
    "sourceIdentity": "MyWebIdentityUser",
    "assumedRoleUser": {
      "assumedRoleId": "AROA123456789EXAMPLE:<assigned role session name>",
      "arn": "arn:aws:sts::444455556666:assumed-role/FederatedWebIdentityRole/<assigned
 role session name>"
    },
    "provider": "arn:aws:iam::444455556666:oidc-provider/<issuer url of OIDC
 provider>",
    "audience": "<id of application>"
  },
  "additionalEventData": {
    "identityProviderConnectionVerificationMethod": "IAMTrustStore"
  },
  "requestID": "aEXAMPLE-0b26-40df-8973-c7012EXAMPLE",
  "eventID": "aEXAMPLE-ee29-4ac0-a0ed-3f5c5EXAMPLE",
  "readOnly": true,
```

```
  "resources": [
    {
      "accountId": "444455556666",
      "type": "AWS::IAM::Role",
      "ARN": "arn:aws:iam::444455556666:role/FederatedWebIdentityRole"
    }
  ],
  "eventType": "AwsApiCall",
  "managementEvent": true,
  "recipientAccountId": "444455556666",
  "eventCategory": "Management",
  "tlsDetails": {
    "tlsVersion": "TLSv1.3",
    "cipherSuite": "TLS_AES_128_GCM_SHA256",
    "clientProvidedHostHeader": "sts.us-east-2.amazonaws.com"
  }
}
```

CloudTrail 日志文件中使用全局端点的示例 Amazon STS 事件

对于向 Amazon Security Token Service (Amazon STS) 全局端点 (`https://sts.amazonaws.com`) 发出的请求，Amazon STS 包括额外的 Amazon CloudTrail 日志字段：`endpointType` 和 `awsServingRegion`。这些字段出现在 `additionalEventData` `RequestDetails` 元素下，用于记录正在调用的服务 Amazon Web Services 区域 区域和端点类型。`endpointType` 字段可以具有 `global` 或 `regional` 的值，以指示处理请求的全局端点的类型。有关 Amazon STS 全局端点更改的更多信息，请参阅 Amazon STS 区域和端点。

> **Note**
>
> 对 Amazon STS 全局端点发出的请求的 Amazon CloudTrail 日志将发送到美国东部（弗吉尼亚州北部）区域。由 Amazon STS 区域端点处理的请求的 CloudTrail 日志将继续记录到 CloudTrail 中的相应区域。

以下示例显示了从欧洲地区（斯德哥尔摩）区域 eu-north-1 向全局端点 (`https://sts.amazonaws.com`) 发出的 Amazon STS 请求的 CloudTrail 日志条目。`endpointType` 字段值为 `global` 表示 Amazon STS 请求由欧洲地区（斯德哥尔摩）区域的全局端点节点处理。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
```

```
        "type": "AssumedRole",
        "principalId": "AROA123456789EXAMPLE:developer",
        "arn": "arn:aws:sts::777788889999:assumed-role/Admin/developer",
        "accountId": "777788889999",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA123456789EXAMPLE",
                "arn": "arn:aws:iam::777788889999:role/Admin",
                "accountId": "777788889999",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2025-02-12T21:44:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-02-12T22:16:48Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/2.15.33 Python/3.11.8 Linux/5.10.233-204.894.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/sts.assume-role",
    "requestParameters": {
        "roleArn": "arn:aws:iam::777788889999:role/test-role",
        "roleSessionName": "test-global-assume-role"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "<encoded session token blob>",
            "expiration": "Feb 12, 2025, 11:16:48 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROA987654321EXAMPLE:test-global-assume-role",
            "arn": "arn:aws:sts::777788889999:assumed-role/test-role/test-global-
assume-role"
        }
    },
    "additionalEventData": {
```

```
        "RequestDetails": {
            "awsServingRegion": "eu-north-1",
            "endpointType": "global"
        }
    },
    "requestID": "EXAMPLE7-2497-457a-9586-f21feEXAMPLE",
    "eventID": "EXAMPLEc-3d26-4c3a-9c94-722a9EXAMPLE",
    "readOnly": true,
    "resources": [
        {
            "accountId": "777788889999",
            "type": "AWS::IAM::Role",
            "ARN": "arn:aws:iam::777788889999:role/test-role"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "777788889999",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "sts-global.eu-north-1.amazonaws.com"
    }
}
```

为了进行比较，以下示例显示了对区域端点 (`https://sts.us-west-2.amazonaws.com`) 发出的 Amazon STS 请求的 CloudTrail 日志条目，该请求由欧洲地区（斯德哥尔摩）区域 eu-north-1 中的区域端点处理。`endpointType` 字段值为 `regional` 表示 Amazon STS 请求由欧洲地区（斯德哥尔摩）区域的全局端点节点处理。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA123456789EXAMPLE:developer",
        "arn": "arn:aws:sts::777788889999:assumed-role/Admin/developer",
        "accountId": "777788889999",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA123456789EXAMPLE",
```

```
                "arn": "arn:aws:iam::777788889999:role/Admin",
                "accountId": "777788889999",
                "userName": "Admin"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2025-02-12T21:44:28Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2025-02-12T22:16:30Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRole",
    "awsRegion": "eu-north-1",
    "sourceIPAddress": "192.0.2.0",
    "userAgent": "aws-cli/2.15.33 Python/3.11.8 Linux/5.10.233-204.894.amzn2int.x86_64
 exe/x86_64.amzn.2 prompt/off command/sts.assume-role",
    "requestParameters": {
        "roleArn": "arn:aws:iam::777788889999:role/test-role",
        "roleSessionName": "test-global-assume-role"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "<encoded session token blob>",
            "expiration": "Feb 12, 2025, 11:16:30 PM"
        },
        "assumedRoleUser": {
            "assumedRoleId": "AROA987654321EXAMPLE:test-global-assume-role",
            "arn": "arn:aws:sts::777788889999:assumed-role/test-role/test-global-
assume-role"
        }
    },
    "additionalEventData": {
        "RequestDetails": {
            "endpointType": "regional",
            "awsServingRegion": "eu-north-1"
        }
    },
    "requestID": "EXAMPLEd-2116-4cd7-bd72-9f72fEXAMPLE",
    "eventID": "EXAMPLEd-219a-48ed-bc54-00e3cEXAMPLE",
    "readOnly": true,
    "resources": [
```

```
        {
            "accountId": "777788889999",
            "type": "AWS::IAM::Role",
            "ARN": "arn:aws:iam::777788889999:role/test-role"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "777788889999",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "sts.eu-north-1.amazonaws.com"
    }
}
```

CloudTrail 日志中的登录事件示例

CloudTrail 日志文件包含的事件采用 JSON 格式。登录事件表示单个登录请求，并包括有关登录主体、区域以及操作的日期和事件的信息。

CloudTrail 日志文件中的登录成功事件示例

以下示例显示了一个成功的登录事件的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/John",
    "accountId": "111122223333",
    "userName": "John"
  },
  "eventTime": "2014-07-16T15:49:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.110",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "requestParameters": null,
```

```
  "responseElements": {
    "ConsoleLogin": "Success"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/s3/"
  },
  "eventID": "3fcfb182-98f8-4744-bd45-10a395ab61cb"
}
```

以下示例显示一个成功授权码请求的 CloudTrail 日志条目。

```
{
"eventVersion": "1.11",
"userIdentity": {
"type": "AssumedRole",
"principalId": "AROATJHQDX737YZP72NTF:thesjain-Isengard",
"arn": "arn:aws:sts::225989345271:assumed-role/Admin/thesjain-Isengard",
"accountId": "225989345271",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "AROATJHQDX737YZP72NTF",
"arn": "arn:aws:iam::225989345271:role/Admin",
"accountId": "225989345271",
"userName": "Admin"
},
"attributes": {
"creationDate": "2025-11-17T22:50:14Z",
"mfaAuthenticated": "false"
}
}
},
"eventTime": "2025-11-17T22:51:32Z",
"eventSource": "signin.amazonaws.com",
"eventName": "AuthorizeOAuth2Access",
"awsRegion": "us-east-1",
"sourceIPAddress": "52.94.133.136",
"userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/142.0.0.0 Safari/537.36",
"requestParameters": {
"scope": "openid",
```

```
"redirect_uri": "http://127.0.0.1:53037/oauth/callback",
"code_challenge_method": "SHA-256",
"client_id": "arn:aws:signin:::devtools/same-device"
},
"responseElements": null,
"additionalEventData": {
"success": "true",
"x-amzn-vpce-id": ""
},
"requestID": "e2854c76-1cba-4360-9fd1-5037b591466b",
"eventID": "59e1720d-3deb-44ff-933d-6828be2a860a",
"readOnly": true,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "225989345271",
"eventCategory": "Management",
"tlsDetails": {
"tlsVersion": "TLSv1.3",
"cipherSuite": "TLS_AES_128_GCM_SHA256",
"clientProvidedHostHeader": "us-east-1.signin.aws.amazon.com"
}
}
```


以下示例显示一个成功 OAuth2 令牌创建请求的 CloudTrail 日志条目。


```
{
"eventVersion": "1.11",
"userIdentity": {
"type": "AssumedRole",
"principalId": "AROATJHQDX737YZP72NTF:jacobjoj-Isengard",
"arn": "arn:aws:sts::225989345271:assumed-role/Admin/jacobjoj-Isengard",
"accountId": "225989345271",
"sessionContext": {
"sessionIssuer": {
"type": "Role",
"principalId": "AROATJHQDX737YZP72NTF",
"arn": "arn:aws:iam::225989345271:role/Admin",
"accountId": "225989345271",
"userName": "Admin"
},
```

```
"attributes": {
"creationDate": "2025-11-18T20:38:10Z",
"mfaAuthenticated": "false"
}
}
},
"eventTime": "2025-11-18T20:38:44Z",
"eventSource": "signin.amazonaws.com",
"eventName": "CreateOAuth2Token",
"awsRegion": "us-east-1",
"sourceIPAddress": "15.248.6.6",
"userAgent": "aws-cli/2.32.0 md/awscrt#0.28.4 ua/2.1 os/macos#24.6.0 md/arch#arm64
 lang/python#3.13.9 md/pyimpl#CPython m/b,AA,Z,E cfg/retry-mode#standard md/
installer#exe sid/35033f4ca1bd md/prompt#off md/command#login",
"requestParameters": {
"client_id": "arn:aws:signin:::devtools/same-device"
},
"responseElements": null,
"additionalEventData": {
"success": "true",
"x-amzn-vpce-id": ""
},
"requestID": "94562943-c85b-4dc1-bf72-43b0fd42d6de",
"eventID": "0b338fac-6a10-4740-b34d-1bb6923e799e",
"readOnly": true,
"eventType": "AwsApiCall",
"managementEvent": true,
"recipientAccountId": "225989345271",
"eventCategory": "Management",
"tlsDetails": {
"tlsVersion": "TLSv1.3",
"cipherSuite": "TLS_AES_128_GCM_SHA256",
"clientProvidedHostHeader": "us-east-1.signin.aws.amazon.com"
}
}
```


有关 CloudTrail 日志文件中所包含信息的更多详细信息，请参阅 Amazon CloudTrail 用户指南中的 CloudTrail 事件参考。

CloudTrail 日志文件中的登录失败事件示例

以下示例显示了一个失败的登录事件的 CloudTrail 日志条目。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
    "principalId": "AIDACKCEVSQ6C2EXAMPLE",
    "arn":"arn:aws:iam::111122223333:user/JaneDoe",
    "accountId": "111122223333",
    "userName": "JaneDoe"
  },
  "eventTime": "2014-07-08T17:35:27Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.100",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "Failed authentication",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "MobileVersion": "No",
    "LoginTo": "https://console.aws.amazon.com/sns"
  },
  "eventID": "11ea990b-4678-4bcd-8fbe-62509088b7cf"
}
```

从此信息，您可以确定此次登录尝试源自名为 JaneDoe 的 IAM 用户，如 userIdentity 元素中所示。您还可以发现此次登录尝试失败，如 responseElements 元素中所示。可以看出 JaneDoe 是在 2014 年 7 月 8 日下午 5:35 (UTC) 尝试登录 Amazon SNS 控制台的。

用户名称不正确导致的登录失败事件示例

以下示例显示了一个失败登录事件的 CloudTrail 日志条目，该事件是由于用户输入错误用户名导致的。Amazon 用 HIDDEN_DUE_TO_SECURITY_REASONS 代替了 userName 文本以帮助防止泄露可能的敏感信息。

```
{
  "eventVersion": "1.05",
  "userIdentity": {
    "type": "IAMUser",
```

```
    "accountId": "123456789012",
    "accessKeyId": "",
    "userName": "HIDDEN_DUE_TO_SECURITY_REASONS"
  },
  "eventTime": "2015-03-31T22:20:42Z",
  "eventSource": "signin.amazonaws.com",
  "eventName": "ConsoleLogin",
  "awsRegion": "us-east-2",
  "sourceIPAddress": "192.0.2.101",
  "userAgent": "Mozilla/5.0 (Windows NT 6.1; WOW64; rv:24.0) Gecko/20100101
 Firefox/24.0",
  "errorMessage": "No username found in supplied account",
  "requestParameters": null,
  "responseElements": {
    "ConsoleLogin": "Failure"
  },
  "additionalEventData": {
    "LoginTo": "https://console.aws.amazon.com/console/home?state=hashArgs
%23&isauthcode=true",
    "MobileVersion": "No"
  },
  "eventID": "a7654656-0417-45c6-9386-ea8231385051",
  "eventType": "AwsConsoleSignin",
  "recipientAccountId": "123456789012"
}
```


IAM 角色信任策略行为

2022 年 9 月 21 日，Amazon 对 IAM 角色信任策略行为进行了更改，要求角色代入自己时在角色信任策略中显式允许。对于 AssumeRole 事件，旧行为允许列表中的 IAM 角色会有一个有关 explicitTrustGrant 的 additionalEventData 字段。当旧允许列表中的角色使用旧行为代入自己时，explicitTrustGrant 的值为 false。当旧允许列表中的角色代入自己，但角色信任策略行为已更新为显式允许该角色代入自己时，explicitTrustGrant 的值为 true。

只有极少数的 IAM 角色出现在旧行为的允许列表中，并且只有当这些角色代入自己时，此字段才会出现在其 CloudTrail 日志中。在大多数情况下，IAM 角色并不需要代入自己。Amazon 建议通过更新进程、代码或配置来消除此行为，或者更新角色信任策略以显式允许这种行为。有关更多信息，请参阅 Announcing an update to IAM role trust policy behavior。

在 Amazon CloudTrail 中跟踪特权任务

IAM 的 Amazon Organizations 管理账户或委派管理员账户可以使用短期根访问权限对成员账户执行一些根用户任务。短期特权会话为您提供临时凭证，您可以限定这些凭证的范围，以对组织中的成员账户执行特权操作。您可以使用以下步骤来标识管理账户或委派管理员在 `sts:AssumeRoot` 会话期间采取的操作。

> **Note**
>
> 全局端点不支持 `sts:AssumeRoot`。CloudTrail 记录为端点指定的区域中的 `ConsoleLogin` 事件。

要在 CloudTrail 日志中跟踪特权会话执行的操作

1. 在您的 CloudTrail 日志中查找 `AssumeRoot` 事件。当您的管理账户或 IAM 的委派管理员从 `sts:AssumeRoot` 中获得一组短期证书时，即会生成此事件。

   在以下示例中，AssumeRoot 的 CloudTrail 事件日志记录在 `eventName` 字段中。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AIDACKCEVSQ6C2EXAMPLE:JohnRole1",
        "arn": "arn:aws:sts::111111111111:assumed-role/John/JohnRole1",
        "accountId": "111111111111",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AIDACKCEVSQ6C2EXAMPLE",
                "arn": "arn:aws:iam::111111111111:role/John",
                "accountId": "111111111111",
                "userName": "Admin2"
            },
            "webIdFederationData": {},
            "attributes": {
                "creationDate": "2024-10-25T20:45:28Z",
                "mfaAuthenticated": "false"
            },
```

```
            "assumedRoot": "true"
        }
    },
    "eventTime": "2024-10-25T20:52:11Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoot",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "requestParameters": {
        "targetPrincipal": "222222222222",
        "taskPolicyArn": {
            "arn": "arn:aws:iam::aws:policy/root-task/S3UnlockBucketPolicy"
        }
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
            "expiration": "Oct 25, 2024, 9:07:11 PM"
        }
    }
}
```

有关访问您的 CloudTrail 日志的步骤，请参阅《Amazon CloudTrail 用户指南》中的 Getting and viewing your CloudTrail log files。

2. 在 CloudTrail 事件日志中，找到指定所采取的成员账户操作的 `targetPrincipal`，以及 `AssumeRoot` 会话的唯一 `accessKeyId`。

   在以下示例中，`targetPrincipal` 是 222222222222，`accessKeyId` 是 ASIAIOSFODNN7EXAMPLE。

```
"eventTime": "2024-10-25T20:52:11Z",
    "eventSource": "sts.amazonaws.com",
    "eventName": "AssumeRoot",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "requestParameters": {
        "targetPrincipal": "222222222222",
        "taskPolicyArn": {
            "arn": "arn:aws:iam::aws:policy/root-task/S3UnlockBucketPolicy"
        }
    },
```

```
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY",
            "expiration": "Oct 25, 2024, 9:07:11 PM"
    }
}
```

3. 在目标主体的 CloudTrail 日志中，搜索与 AssumeRoot 事件中的 accessKeyId 值相对应的访问密钥 ID。使用 eventName 字段值确定 AssumeRoot 会话期间所执行的特权任务。单个会话中可能有多个特权任务在执行。AssumeRoot 的最大会话持续时间为 900 秒（15 分钟）。

   在以下示例中，管理账户或委派管理员删除了 Amazon S3 存储桶的基于资源的策略。

```
{
    "eventVersion": "1.10",
    "userIdentity": {
        "type": "Root",
        "principalId": "222222222222",
        "arn": "arn:aws:iam::222222222222:root",
        "accountId": "222222222222",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "attributes": {
                "creationDate": "2024-10-25T20:52:11Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2024-10-25T20:53:47Z",
    "eventSource": "s3.amazonaws.com",
    "eventName": "DeleteBucketPolicy",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "requestParameters": {
        "bucketName": "resource-policy-John",
        "Host": "resource-policy-John.s3.amazonaws.com",
        "policy": ""
    },
    "responseElements": null,
    "requestID": "1234567890abcdef0",
    "eventID": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
    "readOnly": false,
```

```
    "resources": [
        {
            "accountId": "222222222222",
            "type": "AWS::S3::Bucket",
            "ARN": "arn:aws:s3:::resource-policy-John"
        }
    ],
    "eventType": "AwsApiCall",
    "managementEvent": true,
    "recipientAccountId": "222222222222",
    "eventCategory": "Management",
    "tlsDetails": {
        "tlsVersion": "TLSv1.3",
        "cipherSuite": "TLS_AES_128_GCM_SHA256",
        "clientProvidedHostHeader": "resource-policy-John.s3.amazonaws.com"
    }
}
```


Amazon Identity and Access Management 的合规性验证

作为多项 Amazon 合规性计划的一部分，第三方审计员将评估 Amazon Identity and Access Management (IAM) 的安全性和合规性。这些合规性计划包括 SOC、PCI、FedRAMP、ISO 等。

要了解某个 Amazon Web Services 服务是否在特定合规性计划范围内，请参阅按合规性计划划分的范围内的 Amazon Web Services 服务，然后选择您感兴趣的合规性计划。有关常规信息，请参阅 、、Amazon Web Services 合规性计划。

您可以使用 Amazon Artifact 下载第三方审计报告。有关更多信息，请参阅在 Amazon Artifact 中下载报告。

您在使用 Amazon Web Services 服务 时的合规性责任由您的数据的敏感性、您公司的合规性目标以及适用的法律法规决定。有关您在使用 Amazon Web Services 服务 时的合规责任的更多信息，请参阅 Amazon 安全性文档。

Amazon Identity and Access Management 中的故障恢复能力

Amazon全球基础架构围绕Amazon区域和可用区构建。Amazon区域有多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。有关 Amazon 区域和可用区的更多信息，请参阅 Amazon 全球基础架构。

Amazon Identity and Access Management（IAM）和 Amazon Security Token Service（Amazon STS）是可自我维持、基于区域的服务，可在全球范围内使用。

IAM 是一种非常关键的 Amazon Web Services 服务。在 Amazon 中执行的每项操作必须由 IAM 进行身份验证和授权。IAM 根据存储在 IAM 中的身份和策略检查每个请求，以确定是允许还是拒绝请求。IAM 的设计采用了单独的控制面板和数据面板，这样，即使在意外故障期间，服务也能进行身份验证。授权中使用的 IAM 资源（如角色和策略）存储在控制面板中。IAM 客户可以通过使用 `DeletePolicy` 和 `AttachRolePolicy` 等 IAM 操作来更改这些资源的配置。这些配置更改请求将发送到控制面板。所有商业 Amazon Web Services 区域 都有一个 IAM 控制面板，位于美国东部（弗吉尼亚州北部）区域。随后，IAM 系统会在每个已启用的 Amazon Web Services 区域中将配置更改传播到 IAM 数据面板。IAM 数据面板本质上是 IAM 控制面板配置数据的只读副本。每个 Amazon Web Services 区域 都有一个完全独立的 IAM 数据面板实例，该实例对来自同一区域的请求执行身份验证和授权。在每个区域中，IAM 数据面板至少分布在三个可用区中，并且具有足够的容量来容忍可用区丢失而不会对客户造成任何损害。IAM 控制面板和数据面板都是为零停机时间而构建，所有软件更新和扩展操作均以对客户不可见的方式执行。

在默认启用的区域中，发往 Amazon STS 全局端点的请求会自动在请求发起的同一区域中处理。在选择加入区域中，发往 Amazon STS 全局端点的请求由单个 Amazon Web Services 区域美国东部（弗吉尼亚州北部）处理。但是，您可以选择使用区域 Amazon STS 端点来减少延迟或为应用程序提供额外冗余。要了解更多信息，请参阅管理 Amazon Web Services 区域中的 Amazon STS。

某些事件可能会中断 Amazon Web Services 区域 之间的网络通信。但是，即使您无法与全局 IAM 端点进行通信，Amazon STS 仍然可以对 IAM 主体进行身份验证，并且 IAM 可以授权您的请求。中断通信的事件的具体详细信息将决定您访问 Amazon 服务的能力。在大多数情况下，您可继续在 Amazon 环境中使用 IAM 凭证。以下情况可能适用于中断通信的事件。

IAM 用户的访问密钥

> 您可以在具有长期 IAM 用户的访问密钥的区域中无限期地进行身份验证。当您使用 Amazon Command Line Interface 和 API 时，您需要提供您的 Amazon 访问密钥，以便 Amazon 可以在编程请求中验证您的身份。
>
> **Important**
>
> 作为最佳实践，我们建议您的用户使用临时凭证而不是长期访问密钥来登录。

临时凭证

您可以使用 Amazon STS 区域性服务端点申请新的临时凭证，凭证至少在 24 小时内有效。以下 API 操作会生成临时凭证。

- AssumeRole
- AssumeRoleWithWebIdentity
- AssumeRoleWithSAML
- GetFederationToken
- GetSessionToken

主体和权限

- 您可能无法在 IAM 中添加、修改或删除主体或权限。
- 您的凭证可能无法反映您最近在 IAM 中应用的权限的更改。有关更多信息，请参阅 我所做的更改可能不会立即可见。

Amazon Web Services 管理控制台

- 您可能可以作为 IAM 用户使用区域性登录端点登录 Amazon Web Services 管理控制台。区域性登录端点具有以下 URL 格式。

  ```
  https://{Account ID}.signin.aws.amazon.com/console?region={Region}
  ```

  示例：https://111122223333.signin.aws.amazon.com/console?region=us-west-2
- 您可能无法完成 Universal 2nd Factor（U2F）多重身份验证（MFA）。

IAM 恢复能力的最佳实践

Amazon 已将恢复能力内置于 Amazon Web Services 区域 和可用区。若您在与您的环境交互的系统中观察到有以下 IAM 最佳实践，您就可以利用这种恢复能力。

1. 使用 Amazon STS 区域性服务端点而不是默认的全局端点。
2. 检查您的环境中是否存在例行创建或修改 IAM 资源的重要资源的配置，并准备使用现有 IAM 资源的回退解决方案。

Amazon Identity and Access Management 中的基础结构安全性

作为一项托管式服务，Amazon Identity and Access Management 受 Amazon 全球网络安全保护。有关 Amazon 安全服务以及 Amazon 如何保护基础设施的信息，请参阅 Amazon 云安全性。要按

照基础设施安全最佳实践设计您的 Amazon 环境，请参阅《安全性支柱 Amazon Well-Architected Framework》中的基础设施保护。

您可以使用 Amazon 发布的 API 调用通过网络访问 IAM。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

可使用 IAM HTTPS API（它可让您直接向服务发布 HTTPS 请求）以编程方式访问 IAM。查询 API 将返回敏感信息，包括安全凭证。因此，您必须将 HTTPS 用于所有 API 请求。使用 HTTPS API 时，必须添加代码，才能使用您的凭证对请求进行数字化签名。

您可以从任何网络位置调用这些 API 操作，但 IAM 不支持基于资源的访问策略，其中可以包含基于源 IP 地址的限制。您还可以使用 IAM policy 来控制来自特定 Amazon Virtual Private Cloud (Amazon VPC) 端点或特定 VPC 的访问。事实上，这隔离了在 Amazon 网络中仅从特定 VPC 到给定 IAM 资源的网络访问。

Amazon Identity and Access Management 中的配置和漏洞分析

Amazon 负责处理基本安全任务，如来宾操作系统（OS）和数据库补丁、防火墙配置和灾难恢复等。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下 资源：

- 责任共担模式
- Amazon Web Services：安全过程概述（白皮书）

以下资源还处理 Amazon Identity and Access Management（IAM）中的配置和漏洞分析：

- Amazon Identity and Access Management 的合规性验证
- Amazon Identity and Access Management 中的安全最佳实践和使用案例

适用于 Amazon Identity and Access Management and Access Analyzer 的 Amazon 托管式策略

Amazon 托管式策略是由 Amazon 创建和管理的独立策略。Amazon 托管式策略旨在为许多常见使用案例提供权限，以便您可以开始为用户、组和角色分配权限。

请记住，Amazon 托管式策略可能不会为您的特定使用案例授予最低权限，因为它们可供所有 Amazon 客户使用。我们建议通过定义特定于使用案例的客户管理型策略来进一步减少权限。

您无法更改 Amazon 托管式策略中定义的权限。如果 Amazon 更新在 Amazon 托管式策略中定义的权限，则更新会影响该策略所附加到的所有主体身份（用户、组和角色）。当新的 Amazon Web Services 服务 启动或新的 API 操作可用于现有服务时，Amazon 最有可能更新 Amazon 托管式策略。

有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略。

IAMReadOnlyAccess

使用 `IAMReadOnlyAccess` 托管式策略来允许对 IAM 资源的只读权限。此策略授予权限，使其能够获取和列出所有 IAM 资源。它允许查看用户、组、角色、策略、身份提供程序和 MFA 设备的详细信息和活动报告。它不包括创建或删除资源或访问 IAM Access Analyzer 资源的能力。有关此策略支持的服务和操作的完整列表，请查看此 policy（策略）。

IAMUserChangePassword

使用 `IAMUserChangePassword` 托管式策略可允许 IAM 用户更改他们的密码。

通过配置 IAM Account 设置和 Password 策略，IAM 用户可以修改自己的 IAM 账户密码。当您允许此操作时，IAM 会向每个用户附加该策略。

要查看 JSON 策略，请参阅《Amazon 托管式策略参考指南》中的 IAMUserChangePassword。

IAMAccessAnalyzerFullAccess

使用 `IAMAccessAnalyzerFullAccess` Amazon 托管策略，以允许您的管理员访问 IAM Access Analyzer。

权限分组

此策略根据提供的权限集分为多个语句。

- IAM Access Analyzer — 允许对 IAM Access Analyzer 中所有资源的完全管理权限。

- 创建服务链接角色 — 允许管理员创建服务关联角色，它允许 IAM Access Analyzer 代表您分析其他服务中的资源。此权限允许创建仅供 IAM Access Analyzer 使用的服务关联角色。
- Amazon Organizations — 允许管理员将 IAM Access Analyzer 用于 Amazon Organizations 中的企业。在 Amazon Organizations 为 IAM Access Analyzer 启用可信访问权限后，管理账户的成员可以查看整个企业的结果。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "access-analyzer:*"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "iam:CreateServiceLinkedRole",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "iam:AWSServiceName": "access-analyzer.amazonaws.com"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": [
        "organizations:DescribeAccount",
        "organizations:DescribeOrganization",
        "organizations:DescribeOrganizationalUnit",
        "organizations:ListAccounts",
        "organizations:ListAccountsForParent",
        "organizations:ListAWSServiceAccessForOrganization",
        "organizations:ListChildren",
        "organizations:ListDelegatedAdministrators",
        "organizations:ListOrganizationalUnitsForParent",
        "organizations:ListParents",
```

```
        "organizations:ListRoots"
      ],
      "Resource": "*"
    }
  ]
}
```

IAMAccessAnalyzerReadOnlyAccess

要允许对 IAM Access Analyzer 进行只读访问，请使用 `IAMAccessAnalyzerReadOnlyAccess` Amazon 托管策略。

要同时允许对用于 Amazon Organizations 的 IAM Access Analyzer 进行只读访问中，请创建一个客户托管策略，以允许来自 IAMAccessAnalyzerFullAccess Amazon 托管策略的“Describe”（描述）和“List”（列示）操作。

服务级别权限

此策略提供对 IAM Access Analyzer 的只读访问权限。此策略中不包含其他服务权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "IAMAccessAnalyzerReadOnlyAccess",
      "Effect": "Allow",
      "Action": [
        "access-analyzer:CheckAccessNotGranted",
        "access-analyzer:CheckNoNewAccess",
        "access-analyzer:Get*",
        "access-analyzer:List*",
        "access-analyzer:ValidatePolicy"
      ],
      "Resource": "*"
    }
  ]
}
```

AccessAnalyzerServiceRolePolicy

您无法将 AccessAnalyzerServiceRolePolicy 附加到您的 IAM 实体。此附加到服务相关角色的策略允许 IAM Access Analyzer 代表您执行操作。有关更多信息，请参阅将服务相关角色用于 Amazon Identity and Access Management and Access Analyzer。

权限分组

该策略允许访问 IAM Access Analyzer，以分析来自多个 Amazon Web Services 服务 的资源元数据。

- Amazon DynamoDB – 允许查看 DynamoDB 流和表的权限。
- Amazon Elastic Compute Cloud – 允许描述 IP 地址、快照和 VPC 的权限。
- Amazon Elastic Container Registry – 允许描述图像存储库、检索账户设置并检索注册表和存储库策略。
- Amazon Elastic File System – 允许查看 Amazon EFS 文件系统的描述和查看 Amazon EFS 文件系统的资源级策略的权限。
- Amazon Identity and Access Management – 允许检索有关指定角色的信息并列出具有指定路径前缀的 IAM 角色的权限。允许检索用户、IAM 组、登录配置文件、访问密钥和上次访问服务数据相关信息的权限。
- Amazon Key Management Service – 允许查看有关 KMS 密钥及其密钥策略和授权的详细信息的权限。
- Amazon Lambda – 允许查看有关 Lambda 别名、函数、层和别名信息的权限。
- Amazon Organizations：允许对 Amazon Organizations 的权限，并允许在作为信任区域的 Amazon 组织中创建分析程序。
- Amazon Relational Database Service – 允许查看有关 Amazon RDS 数据库快照和 Amazon RDS 数据库集群快照的详细信息的权限。
- Amazon Simple Storage Service：允许查看有关 Amazon S3 接入点、存储桶、Amazon S3 存储桶接入点和目录存储桶的详细信息的权限。
- Amazon Secrets Manager – 允许查看有关密钥和附加到密钥的资源策略详细信息的权限。
- Amazon Simple Notification Service – 允许查看有关某个主题的详细信息的权限。
- Amazon Simple Queue Service – 允许查看有关指定队列的详细信息的权限。

要查看 JSON 策略，请参阅《Amazon 托管式策略参考指南》中的 AccessAnalyzerServiceRolePolicy。

IAMAuditRootUserCredentials

在对 Amazon Organizations 成员账户执行特权任务以审计成员账户的根用户凭证状态时，使用 IAMAuditRootUserCredentials Amazon 托管策略缩小权限范围。您可以列出或获取个人根用户凭证信息，例如：

- 是否有根用户密码
- 根用户是否有访问密钥以及上次使用该密钥的时间
- 如果根用户关联了签名证书
- 根用户关联的 MFA 设备
- 统一的根用户凭证状态列表

您不能将 IAMAuditRootUserCredentials 附加到自己的 IAM 实体。此策略附加到 AssumeRoot，用于对组织中的成员账户执行特权任务。有关更多信息，请参阅 集中管理成员账户的根访问权限。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyAllOtherActionsOnAnyResource",
         "NotAction": [
            "iam:ListAccessKeys",
            "iam:ListSigningCertificates",
            "iam:GetLoginProfile",
            "iam:ListMFADevices",
            "iam:GetAccountSummary",
            "iam:GetUser",
            "iam:GetAccessKeyLastUsed"
         ],
         "Effect": "Deny",
         "Resource": "*"
      },
      {
         "Sid": "DenyAuditingCredentialsOnNonRootUserResource",
         "Action": [
            "iam:ListAccessKeys",
```

```
            "iam:ListSigningCertificates",
            "iam:GetLoginProfile",
            "iam:ListMFADevices" ,
            "iam:GetUser",
            "iam:GetAccessKeyLastUsed"
         ],
         "Effect": "Deny",
         "NotResource": "arn:aws:iam::*:root"
      }
   ]
}
```

权限分组

此策略根据提供的权限集分为多个语句。

- DenyAllOtherActionsOnAnyResource – 拒绝访问所有资源的凭证。
- DenyAuditingCredentialsOnNonRootUserResource – 拒绝访问所有非根用户资源的凭证。

IAMCreateRootUserPassword

在对 Amazon Organizations 成员账户执行特权任务以允许恢复没有根用户凭证的成员账户的密码时，使用 `IAMCreateRootUserPassword` Amazon 托管策略缩小权限范围。

您不能将 `IAMCreateRootUserPassword` 附加到自己的 IAM 实体。此策略附加到 AssumeRoot，用于对组织中的成员账户执行特权任务。有关更多信息，请参阅 集中管理成员账户的根访问权限。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyAllOtherActionsOnAnyResource",
         "NotAction": [
            "iam:CreateLoginProfile",
            "iam:GetLoginProfile"
         ],
         "Effect": "Deny",
         "Resource": "*"
```

```
      },
      {
         "Sid": "DenyCreatingPasswordOnNonRootUserResource",
         "Action": [
            "iam:CreateLoginProfile",
            "iam:GetLoginProfile"
         ],
         "Effect": "Deny",
         "NotResource": "arn:aws:iam::*:root"
      }
   ]
}
```

权限分组

此策略根据提供的权限集分为多个语句。

- DenyAllOtherActionsOnAnyResource – 拒绝访问以获取或创建所有资源的密码。
- DenyCreatingPasswordOnNonRootUserResource – 拒绝访问以获取或创建所有非根用户资源的密码。

IAMDeleteRootUserCredentials

当您对 Amazon Organizations 成员账户执行特权任务以移除根用户凭证（包括密码、访问密钥、签名证书和停用 MFA）时，使用 `IAMDeleteRootUserCredentials` Amazon 托管策略缩小权限范围。此特权操作需要其他权限，以便您可以查看上次使用的凭证信息、验证成员账户根用户上次使用的信息，以及列出要删除的所有根用户凭证的权限。

您不能将 `IAMDeleteRootUserCredentials` 附加到自己的 IAM 实体。此策略附加到 AssumeRoot，用于对组织中的成员账户执行特权任务。有关更多信息，请参阅 集中管理成员账户的根访问权限。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": [
  {
   "Sid": "DenyAllOtherActionsOnAnyResource",
```

```
    "Effect": "Deny",
    "NotAction": [
     "iam:DeleteAccessKey",
     "iam:DeleteSigningCertificate",
     "iam:DeleteLoginProfile",
     "iam:DeactivateMFADevice",
     "iam:ListAccessKeys",
     "iam:ListSigningCertificates",
     "iam:GetLoginProfile",
     "iam:ListMFADevices",
     "iam:GetUser",
     "iam:GetAccessKeyLastUsed"
    ],

    "Resource": "*"
   },
   {
    "Sid": "DenyDeletingRootUserCredentialsOnNonRootUserResource",
    "Effect": "Deny",
    "Action": [
     "iam:DeleteAccessKey",
     "iam:DeleteSigningCertificate",
     "iam:DeleteLoginProfile",
     "iam:DeactivateMFADevice",
     "iam:ListAccessKeys",
     "iam:ListSigningCertificates",
     "iam:GetLoginProfile",
     "iam:ListMFADevices",
     "iam:GetUser",
     "iam:GetAccessKeyLastUsed"
    ],

    "NotResource": "arn:aws:iam::*:root"
   }
  ]
}
```

权限分组

此策略根据提供的权限集分为多个语句。

- DenyAllOtherActionsOnAnyResource – 拒绝访问以获取或删除所有资源的凭证。

- DenyDeletingRootUserCredentialsOnNonRootUserResource – 拒绝访问以获取或删除所有非根用户资源的凭证。

S3UnlockBucketPolicy

当您对 Amazon Organizations 成员账户执行特权任务以移除拒绝所有主体访问 Amazon S3 存储桶的错误配置存储桶策略时，使用 S3UnlockBucketPolicy Amazon 托管策略缩小权限范围。

您不能将 S3UnlockBucketPolicy 附加到自己的 IAM 实体。此策略附加到 AssumeRoot，用于对组织中的成员账户执行特权任务。有关更多信息，请参阅 集中管理成员账户的根访问权限。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [
      {
         "Sid": "DenyAllOtherActionsOnAnyResource",
         "NotAction": [
            "s3:DeleteBucketPolicy",
            "s3:PutBucketPolicy",
            "s3:GetBucketPolicy",
            "s3:ListAllMyBuckets"
         ],
         "Effect": "Deny",
         "Resource": "*"
      },
      {
         "Sid": "DenyManagingBucketPolicyForNonRootCallers",
         "Action": [
            "s3:DeleteBucketPolicy",
            "s3:PutBucketPolicy",
            "s3:GetBucketPolicy",
            "s3:ListAllMyBuckets"
         ],
         "Effect": "Deny",
         "Resource": "*",
         "Condition" : {
            "ArnNotLike" : {
               "aws:PrincipalArn" : "arn:aws:iam::*:root"
            }
```

```
        }
      }
   ]
}
```

权限分组

此策略根据提供的权限集分为多个语句。

- DenyAllOtherActionsOnAnyResource – 拒绝访问所有资源的存储桶策略。
- DenyManagingBucketPolicyForNonRootCallers – 拒绝访问所有非根用户资源的存储桶策略。

SQSUnlockQueuePolicy

当您对 Amazon Organizations 成员账户执行特权任务以删除拒绝所有主体访问 Amazon SQS 队列的 Amazon Simple Queue Service 基于资源的策略时，使用 `SQSUnlockQueuePolicy` Amazon 托管策略缩小权限范围。

您不能将 `SQSUnlockQueuePolicy` 附加到自己的 IAM 实体。此策略附加到 AssumeRoot，用于对组织中的成员账户执行特权任务。有关更多信息，请参阅 集中管理成员账户的根访问权限。

权限分组

此策略根据提供的权限集分为多个语句。

- DenyAllOtherActionsOnAnyResource – 拒绝访问所有资源的 Amazon SQS 操作。
- DenyGettingQueueAttributesOnNonOwnQueue – 拒绝访问另一个账户所拥有的队列的 Amazon SQS 队列属性。
- DenyActionsForNonRootUser – 拒绝访问所有非根用户资源的 Amazon SQS 操作。

IAM 和 IAM Access Analyzer 更新 Amazon 托管策略

查看从服务开始跟踪这些更改以来，有关 IAM 和 Amazon 托管式策略更新的详细信息。有关此页面更改的自动警报，请订阅 IAM 和 IAM Access Analyzer 文档历史记录页面上的 RSS 源。

更改	描述	日期
IAMUserChangePassword – 已添加权限	IAM 向 `IAMUserChangePassword` 添加了权限，以允许在路径内指定的用户。	2025 年 5 月 28 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 添加了对 `AccessAnalyzerServiceRolePolicy` 的服务级别权限的 `iam:GetAccountAuthorizationDetails` 。	2024 年 5 月 12 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 `AccessAnalyzerServiceRolePolicy` 的服务级别权限添加了对 Amazon S3 目录存储桶接入点的支持。	2025 年 3 月 31 日
IAMDeleteRootUserCredentials – 移除的权限	IAM 从托管策略中移除了 `iam:DeleteVirtualMFADevice` 权限。	2025 年 1 月 7 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 `AccessAnalyzerServiceRolePolicy` 的服务级别权限添加了对检索 Amazon ECR 账户设置和注册表策略信息的权限的支持。	2024 年 12 月 10 日
IAMAuditRootUserCredentials – 已添加托管策略	IAM 为集中管理成员账户的根访问添加了托管策略，以限定您可以对 Amazon Organizations 成员账户执行的特权任务的范围。	2024 年 11 月 14 日

更改	描述	日期
IAMCreateRootUserPassword – 已添加托管策略	IAM 为集中管理成员账户的根访问添加了托管策略，以限定您可以对 Amazon Organizations 成员账户执行的特权任务的范围。	2024 年 11 月 14 日
IAMDeleteRootUserCredentials – 已添加托管策略	IAM 为集中管理成员账户的根访问添加了托管策略，以限定您可以对 Amazon Organizations 成员账户执行的特权任务的范围。	2024 年 11 月 14 日
S3UnlockBucketPolicy – 已添加托管策略	IAM 为集中管理成员账户的根访问添加了托管策略，以限定您可以对 Amazon Organizations 成员账户执行的特权任务的范围。	2024 年 11 月 14 日
SQSUnlockQueuePolicy – 已添加托管策略	IAM 为集中管理成员账户的根访问添加了托管策略，以限定您可以对 Amazon Organizations 成员账户执行的特权任务的范围。	2024 年 11 月 14 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 `AccessAnalyzerServiceRolePolicy` 的服务级别权限添加了对检索 IAM 用户和角色标签信息的权限的支持。	2024 年 10 月 29 日

更改	描述	日期
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对检索 IAM 用户和角色策略信息的权限的支持。	2024 年 5 月 30 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对检索阻止公开访问 Amazon EC2 快照当前状态权限的支持。	2024 年 1 月 23 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对 DynamoDB 流和表的支持。	2024 年 1 月 11 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对 Amazon S3 目录存储桶的支持。	2023 年 12 月 1 日
IAMAccessAnalyzerReadOnlyAccess – 添加权限	IAM Access Analyzer 添加了权限，让您可以检查策略更新是否授予额外的访问权限。 IAM Access Analyzer 需要此权限才能对您的策略执行策略检查。	2023 年 11 月 26 日

更改	描述	日期
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 将 IAM 操作添加到 `AccessAnalyzerServiceRolePolicy` 的服务级别权限，以支持以下操作： • 列出策略的实体 • 生成服务上次访问的详细信息 • 列出访问密钥信息	2023 年 11 月 26 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 在 `AccessAnalyzerServiceRolePolicy` 的服务级权限中附加支持以下资源类型： • Amazon EBS 卷快照 • Amazon ECR 存储库 • Amazon EFS 文件系统 • Amazon RDS 数据库快照 • Amazon RDS 数据库集群快照 • Amazon SNS 主题	2022 年 10 月 25 日
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 添加了对 `AccessAnalyzerServiceRolePolicy` 的服务级别权限的 `lambda:GetFunctionUrlConfig` 操作。	2022 年 4 月 6 日

更改	描述	日期
AccessAnalyzerServiceRolePolicy – 添加权限	IAM Access Analyzer 添加了新的 Amazon S3 操作来分析与多区域访问点关联的元数据。	2021 年 9 月 2 日
IAMAccessAnalyzerReadOnlyAccess – 添加权限	IAM Access Analyzer 添加了一个新操作以授予 `ValidatePolicy` 权限，以允许您使用策略检查进行验证。 IAM Access Analyzer 需要此权限才能对您的策略执行策略检查。	2021 年 3 月 16 日
IAM Access Analyzer 开始跟踪更改	IAM Access Analyzer 开始为其 Amazon 托管策略跟踪更改。	2021 年 3 月 1 日

IAM 外部的安全功能

通过 IAM 可以控制对使用 Amazon Web Services 管理控制台、Amazon 命令行工具或服务 API 操作（通过使用 Amazon 开发工具包）执行的任务的访问。某些 Amazon 产品还有其他方法来保护其资源。以下列表提供了一些示例，不过并不详尽。

Amazon EC2

在 Amazon Elastic Compute Cloud 中，需要使用密钥对（对于 Linux 实例）或使用用户名称和密码（对于 Windows 实例）来登录实例。

有关更多信息，请参阅以下文档：

- 《Amazon EC2 用户指南》中的 Amazon EC2 Linux 实例入门
- 《Amazon EC2 用户指南》中的 Amazon EC2 Windows 实例入门

Amazon RDS

在 Amazon Relational Database Service 中，需要使用与数据库关联的用户名称和密码来登录数据库引擎。

有关更多信息，请参阅 Amazon RDS 用户指南 中的 Amazon RDS 入门。

Amazon EC2 和 Amazon RDS

在 Amazon EC2 和 Amazon RDS 中，需要使用安全组来控制发送到实例或数据库的流量。

有关更多信息，请参阅以下文档：

- 《Amazon EC2 用户指南》中的适用于 Linux 实例的 Amazon EC2 安全组
- 《Amazon EC2 用户指南》中的适用于 Windows 实例的 Amazon EC2 安全组
- Amazon RDS 用户指南中的 Amazon RDS 安全组

WorkSpaces

在 Amazon WorkSpaces 中，用户使用用户名称和密码登录桌面。

有关更多信息，请参阅 Amazon WorkSpaces 管理指南中的 WorkSpaces 入门。

这些访问控制方法不是 IAM 的一部分。IAM 允许您控制管理这些 Amazon 产品的方式，方法包括创建或终止 Amazon EC2 实例、设置新 WorkSpaces 桌面等。也就是说，IAM 可帮助您控制通过向 Amazon Web Services 进行请求来执行的任务，并且可帮助您控制对 Amazon Web Services 管理控制台 的访问。但是，IAM 不会帮助您管理诸如登录操作系统（Amazon EC2）、数据库（Amazon RDS）或桌面（Amazon WorkSpaces）等任务的安全性。

当您使用特定 Amazon 产品时，请务必阅读相应文档，了解属于该产品的所有资源的安全选项。

使用 Amazon Identity and Access Management Access Analyzer

Amazon Identity and Access Management Access Analyzer 提供以下功能：

- IAM Access Analyzer 外部访问分析器可帮助您识别组织中的资源以及与外部实体共享的账户。
- IAM Access Analyzer 内部访问分析器有助于识别对您的组织和账户内选定资源的内部访问。
- IAM Access Analyzer 未使用的访问分析器可帮助您识别组织和账户中未使用的访问。
- IAM Access Analyzer 将根据策略语法和 Amazon 最佳实践验证 IAM policy。
- IAM Access Analyzer 自定义策略检查可帮助您根据指定的安全标准验证 IAM policy。
- IAM Access Analyzer 将根据您的 Amazon CloudTrail 日志中的访问活动生成 IAM policy。

识别与外部实体共享的资源

IAM Access Analyzer 帮助您标识企业和账户中与外部实体共享的资源，例如 Amazon S3 存储桶或 IAM 角色。这可以帮助您识别对资源和数据的意外访问，此类访问会带来安全风险。IAM Access Analyzer 使用基于逻辑的推理来分析 Amazon 环境中基于资源的策略，确定与外部主体共享的资源。对于在您的账户外共享的资源的每个实例，IAM Access Analyzer 都会生成一个调查结果。调查发现包括有关访问权限以及该访问权限授予到的外部主体的信息。您可以查看调查发现，以确定该访问权限是否按计划授予且安全，或者该访问权限是否是计划外的访问权限并存在安全风险。除了帮助您识别与外部实体共享的资源外，您还可以使用 IAM Access Analyzer 结果预览策略如何影响对资源的公共和跨账户访问，然后再部署资源权限。调查发现整理到可视化摘要控制面板中。控制面板会突出显示公共和跨账户存取调查发现之间的差异，并按资源类型提供调查发现的明细。要了解有关控制面板的更多信息，请参阅 查看 IAM Access Analyzer 的调查发现控制面板。

> **Note**
>
> 外部实体可以是另一个 Amazon 账户、根用户、IAM 用户或角色、联合身份用户、匿名用户或可用于创建筛选器的另一个实体。有关更多信息，请参阅 Amazon JSON 策略元素：主体。

启用 IAM Access Analyzer 后，可以为整个企业或账户创建分析器。您选择的组织或账户称为分析器的信任区。该分析器将监控信任区域内所有受支持的资源。信任区域内的主体对资源的任何访问都被视为受信任的。启用后，IAM Access Analyzer 将分析应用于您的信任区中所有受支持的资源的策略。

在进行第一次分析后，IAM Access Analyzer 将定期分析这些策略。如果添加了新策略或更改了现有策略，IAM Access Analyzer 会在约 30 分钟内分析新策略或更新后的策略。

在分析策略时，如果 IAM Access Analyzer 发现一个向不在信任区域内的外部主体授予访问权限的策略，则会生成一个结果。每个结果均包含有关资源、有权访问该资源的外部实体以及授予的权限的详细信息，以便您能采取适当的操作。您可以查看查找结果中包含的详细信息，以确定资源访问是有意的还是应解决的潜在风险。在向资源添加策略或更新现有策略时，IAM Access Analyzer 将对策略进行分析。IAM Access Analyzer 还会定期分析所有基于资源的策略。

在某些情况下，IAM Access Analyzer 很少会收到添加或更新了策略的通知，这可能导致生成的调查发现延迟。如果您创建或删除与 Amazon S3 存储桶关联的多区域接入点，或者更新多区域接入点的策略，IAM Access Analyzer 可能需要长达 6 小时才能生成或解析调查发现。此外，如果 Amazon CloudTrail 日志传输或资源控制策略（RCP）限制更改存在传输问题，则策略更改不会触发对调查发现中报告的资源进行重新扫描。在发生此情况时，IAM Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。如果要确认对策略所做的更改是否可以解决查找结果中报告的访问问题，您可以通过使用 Finding（结果）详情页面中的 Rescan（重新扫描）链接，或通过使用 IAM Access Analyzer API 的 `StartResourceScan` 操作，以重新扫描查找结果中报告的资源。要了解更多信息，请参阅解决 IAM Access Analyzer 调查发现。

> **Important**
>
> 对于外部访问，IAM Access Analyzer 仅分析应用于与启用该功能所处同一 Amazon 区域中资源的策略。要监控 Amazon 环境中的所有资源，您必须创建一个外部访问分析器，以便在您使用受支持的 Amazon 资源的每个区域中启用 IAM Access Analyzer。
> 对于未使用的访问，分析器的调查发现不会因区域而变化。无需在您拥有资源的每个区域创建未使用的访问分析器。

IAM Access Analyzer 分析以下资源类型的外部访问：

- Amazon Simple Storage Service 存储桶
- Amazon Simple Storage Service 目录存储桶
- Amazon Identity and Access Management 角色
- Amazon Key Management Service 密钥
- Amazon Lambda 函数和层
- Amazon Simple Queue Service 队列
- Amazon Secrets Manager 密钥

- Amazon Simple Notification Service 主题
- Amazon Elastic Block Store 卷和快照
- Amazon Relational Database Service 数据库快照
- Amazon Relational Database Service 数据库集群快照
- Amazon Elastic Container Registry 存储库
- Amazon Elastic File System 文件系统
- Amazon DynamoDB Streams
- Amazon DynamoDB 表

识别对关键业务资源的内部访问

对于所选关键业务资源，IAM Access Analyzer 帮助识别您的组织或账户内的哪些主体有权访问它们。此分析支持实施最低权限原则，确保只有您的组织内的预期主体才能访问您指定的资源。

内部访问分析可帮助您：

- 确定您账户或组织内的哪些 IAM 用户或角色可以访问指定的资源
- 了解您的 Amazon 环境中主体和资源之间的访问路径
- 验证您的访问控制是否按预期工作
- 查看调查发现，以确定该访问是否是预期且安全的，或者该访问是否是意外访问且存在安全风险
- 识别并修复组织内的意外访问

IAM Access Analyzer 分析以下资源类型的内部访问：

- Amazon Simple Storage Service 存储桶
- Amazon Simple Storage Service 目录存储桶
- Amazon Relational Database Service 数据库快照
- Amazon Relational Database Service 数据库集群快照
- Amazon DynamoDB Streams
- Amazon DynamoDB 表

识别授予 IAM 用户和角色的未使用访问权限

IAM Access Analyzer 可帮助您识别和查看 Amazon 组织和账户中未使用的访问。IAM Access Analyzer 会持续监控 Amazon 组织和账户中的所有 IAM 角色和用户，并生成未使用的访问的调查发现。调查发现会突出显示未使用的角色、IAM 用户未使用的访问密钥以及 IAM 用户未使用的密码。对于活动的 IAM 角色和用户，这些调查发现提供了对未使用的服务和操作的可见性。

IAM Access Analyzer 会查看 Amazon 组织和账户中所有角色的上次访问信息，以帮助您识别未使用的访问。IAM 操作上次访问的信息可帮助您识别 Amazon Web Services 账户 中角色的未使用操作。有关更多信息，请参阅 使用上次访问的信息优化 Amazon 中的权限。

外部、内部和未使用访问分析器的调查发现整理到可视化摘要控制面板中。控制面板会突出显示调查发现最多的 Amazon 资源和 Amazon Web Services 账户，并按类型提供调查发现明细。有关控制面板的更多信息，请参阅 查看 IAM Access Analyzer 的调查发现控制面板。

根据 Amazon 最佳实践验证策略

您可以使用 IAM Access Analyzer 策略验证提供的基本策略检查，根据 IAM policy 语法和 Amazon 最佳实践验证您的策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些调查发现提供了可行的建议，可帮助您编写实用且符合 Amazon 最佳实践的策略。要了解有关使用策略验证来验证策略的更多信息，请参阅 使用 IAM Access Analyzer 验证策略。

根据指定的安全标准验证策略

您可以使用 IAM Access Analyzer 自定义策略检查，根据指定的安全标准验证策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。通过控制台，您可以检查与现有版本相比，更新后的策略是否授予新的访问权限。通过 Amazon CLI 和 Amazon API，您还可以检查策略不允许您认为关键的特定 IAM 操作。这些检查突出显示了授予新访问权限的策略语句。您可以更新策略语句并重新运行检查，直到策略符合您的安全标准。要了解有关使用自定义策略检查验证策略的更多信息，请参阅 使用 IAM Access Analyzer 自定义策略检查来验证策略。

生成策略

IAM Access Analyzer 分析 Amazon CloudTrail 日志以识别指定日期范围内 IAM 实体（用户或角色）已使用的操作和服务。然后，它会生成基于该访问活动的 IAM policy。您可以使用生成的策略通过将实体的权限附加到 IAM 用户或角色来优化实体权限。要了解有关使用访问 IAM Access Analyzer 生成策略的详细信息，请参阅 IAM Access Analyzer 策略生成。

IAM Access Analyzer 定价

IAM Access Analyzer 根据每月每个分析器分析的 IAM 角色和用户数量对未使用的访问分析收费。

- 您需要为创建的每个未使用的访问分析器付费。
- 跨多个区域创建未使用的访问分析器使您需要为每个分析器付费。
- 不会对服务相关角色未使用的访问活动进行分析，也不会将其包含在分析的 IAM 角色总数中。

IAM Access Analyzer 根据每个内部访问分析器每月监控的资源数量对内部访问分析收费。

IAM Access Analyzer 根据向其发出的用于检查新访问的 API 请求数，对自定义策略检查收费。

有关 IAM Access Analyzer 的收费和价格的完整列表，请参阅 IAM Access Analyzer 定价。

若要查看您的账单，请转到 Amazon 账单与成本管理 控制台中的账单和成本管理控制面板。您的账单中包含了提供您的账单详情的使用情况报告的链接。要了解关于 Amazon Web Services 账户 账单的更多信息，请参阅 Amazon Billing 用户指南。

如果您有关于 Amazon 账单、账户和事件的问题，请联系 Amazon Web Services 支持。

IAM Access Analyzer 调查发现

IAM Access Analyzer 会生成 Amazon Web Services 账户 或组织中的外部访问、内部访问和未使用访问的调查发现。

对于外部访问，IAM Access Analyzer 为基于资源的策略的每个实例生成一个调查发现，该策略可向不在信任区域内的主体授予对信任区域内的资源的访问权限。创建外部访问分析器时，您可以选择组织或 Amazon Web Services 账户 进行分析。您为分析器选择的组织或账户中的任何主体都被视为受信任。由于同一组织或账户中的主体是受信任的，因此组织或账户中的资源和主体构成了分析器的信任区域。信任区域内的任何共享都被视为是安全的，因此 IAM Access Analyzer 不会生成结果。例如，如果选择某个组织作为分析器的信任区域，则该组织中的所有资源和主体都在信任区域内。如果您将某个组织成员账户中的 Amazon S3 存储桶的权限授予其他组织成员账户中的主体，IAM Access Analyzer 不会生成调查发现。但是，如果您向非企业成员账户中的主体授予权限，则 IAM Access Analyzer 会生成结果。

对于内部访问，当您的组织内的 IAM 角色或用户与您指定的资源之间存在可能的访问路径时，IAM Access Analyzer 会生成调查发现。与外部访问分析类似，您选择的范围（组织或账户）决定了什么被视为内部。如果选择一个组织作为范围，IAM Access Analyzer 将生成组织内主体和资源之间的访问

路径的调查发现。如果您选择一个账户，则将为该特定账户内的访问路径生成调查发现。IAM Access Analyzer 使用自动推理来评估所有 IAM 策略，以监控谁有权访问您的资源。

将外部和内部访问调查发现与相同的信任区域相结合，可以对来自定义的信任边界内部和外部的特定资源所有可能访问进行全面分析。

对于未使用访问，IAM Access Analyzer 会为 Amazon 组织和账户中授予的未使用访问生成调查发现。创建未使用的访问分析器时，IAM Access Analyzer 会持续监控 Amazon 组织和账户中的所有 IAM 角色和用户，并生成未使用的访问的调查发现。IAM Access Analyzer 为未使用的访问生成以下类型的调查发现：

- 未使用的角色：在指定使用窗口内没有访问活动的角色。
- 未使用的 IAM 用户访问密钥和密码：未用于在指定使用窗口内访问 Amazon Web Services 账户 的属于 IAM 用户的凭证。
- 未使用的权限：角色在指定使用窗口内未使用的服务级别和操作级别权限。IAM Access Analyzer 使用附加到角色的基于身份的策略来确定这些角色可以访问的服务和操作。IAM Access Analyzer 支持查看所有服务级别权限的未使用权限。有关未使用的访问调查发现支持的操作级别权限的完整列表，请参阅 上次访问 IAM 操作信息的服务和操作。

> **Note**
>
> IAM Access Analyzer 免费提供外部访问调查发现。根据每个分析器每月分析的 IAM 角色和用户数量，对未使用访问调查发现收取费用。根据每个分析器每月监控的 Amazon 资源数量，还会对内部访问调查发现收取费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

主题

- 了解 IAM Access Analyzer 调查发现的工作原理
- Amazon Identity and Access Management Access Analyzer 入门
- 查看 IAM Access Analyzer 的调查发现控制面板
- 查看 IAM Access Analyzer 调查发现
- 筛选 IAM Access Analyzer 调查发现
- 存档 IAM Access Analyzer 调查发现
- 解决 IAM Access Analyzer 调查发现

- IAM Access Analyzer 错误调查发现
- IAM Access Analyzer 支持的外部和内部访问资源类型
- IAM Access Analyzer 的委托管理员。
- 存档规则
- 使用 Amazon EventBridge 监控 Amazon Identity and Access Management Access Analyzer
- 将 IAM Access Analyzer 与 Amazon Security Hub CSPM 集成
- 使用 Amazon CloudTrail 记录 IAM Access Analyzer API 调用
- IAM Access Analyzer 筛选条件键
- 将服务关联角色用于 Amazon Identity and Access Management Access Analyzer

了解 IAM Access Analyzer 调查发现的工作原理

本主题介绍了 IAM Access Analyzer 中用于帮助您熟悉访问分析器如何监控对 Amazon 资源的访问的概念和术语。

外部访问调查发现

对于在信任区域之外共享的每个资源实例，仅生成一次外部访问调查发现。每当修改基于资源的策略时，IAM Access Analyzer 都会分析策略。如果更新的策略共享结果中已标识的资源，但具有不同的权限或条件，则会为该资源共享实例生成新结果。影响资源控制策略（RCP）限制的资源控制策略更改也会生成新的调查发现。IAM Access Analyzer 还会评估由声明性策略建立的访问控制配置。如果第一个调查发现中的访问被删除，则该调查发现的状态将更新为已解决。

所有调查发现保持活动状态，直到您将其存档或删除生成调查发现的访问。在删除访问时，调查发现状态将更新为已解决。

> **Note**
>
> 修改策略后，IAM Access Analyzer 可能需要 30 分钟来分析资源，然后更新外部访问调查发现。资源控制策略（RCP）更改不会触发对调查发现中报告的资源进行重新扫描。IAM Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。

IAM Access Analyzer 如何生成外部访问的调查发现

Amazon Identity and Access Management Access Analyzer 使用一种名为 Zelkova 的技术来分析 IAM 策略并识别对资源的外部访问。

Zelkova 将 IAM 策略转换为等效逻辑语句，并通过一套通用和专用逻辑求解器（可满足性模理论）运行它们。IAM Access Analyzer 反复将 Zelkova 应用于策略，使用越来越具体的查询来根据策略内容表征策略允许的访问类型。有关可满足性模理论的更多信息，请参阅可满足性模理论。

对于外部访问分析器，IAM Access Analyzer 不会检查访问日志来确定外部实体是否实际访问了您信任区域内的资源。相反，当基于资源的策略允许访问资源时，它会生成调查发现，无论该资源是否被外部实体访问。

此外，IAM Access Analyzer 在做出决定时不会考虑任何外部账户的状态。如果其指示账户 111122223333 可以访问您的 Amazon S3 存储桶，则它没有有关该账户中的用户、角色、服务控制策略（SCP）或其他相关配置的任何信息。这是为了保护客户隐私，因为 IAM Access Analyzer 不知道谁拥有其他账户。这也是为了安全起见，因为即使当前没有可以使用它的活跃主体，了解潜在的外部访问也很重要。

IAM Access Analyzer 仅考虑外部用户不能直接影响的或对授权有影响的某些 IAM 条件键。有关 IAM Access Analyzer 考虑的条件键的示例，请参阅 IAM Access Analyzer 筛选条件键。

IAM Access Analyzer 当前不会报告来自 Amazon Web Services 服务 主体或内部服务账户的调查发现。在极少数情况下，它无法完全确定策略语句是否授予对外部实体的访问权限，它会错误地声明一个假阳性结果。这是因为 IAM Access Analyzer 旨在提供您账户中资源共享的全面视图，并减少假阴性结果。

内部访问调查发现

要使用内部访问分析，您必须首先通过选择要监控的特定资源来配置分析器。配置后，当您的组织或账户内的主体（IAM 用户或角色）有权访问所选资源时，会生成内部访问调查发现。当分析器下次扫描指定资源并识别有权访问该资源的主体时，会生成新的调查发现。如果更新后的策略允许已在调查发现中识别但具有不同权限或条件的主体，则会为该主体和资源实例生成新的调查发现。此更新后的策略可以是基于资源的策略、基于身份的策略、服务控制策略 (SCP) 或资源控制策略 (RCP)。IAM Access Analyzer 还会评估由声明性策略建立的访问控制配置。

> **Note**
>
> 内部访问调查发现只能通过 ListFindingsV2 API 操作获得。

IAM Access Analyzer 如何生成内部访问的调查发现

要分析内部访问，即使您已经创建了一个分析器来生成外部访问调查发现或未使用的访问调查发现，也必须为资源的内部访问调查发现创建一个单独的分析器。

创建内部访问分析器后，IAM Access Analyzer 会评估指定账户或组织内的所有基于资源的策略、基于身份的策略、服务控制策略 (SCP)、资源控制策略 (RCP) 和权限边界。

通过创建专用于内部访问选定资源的分析器，您可以识别：

- 您的组织或账户内的主体何时可以访问选定资源
- 根据所有适用策略的交集，主体允许的总有效权限
- 复杂的访问路径，其中主体根据身份策略和资源策略的组合获得访问权限

> **Note**
>
> IAM Access Analyzer 无法为包含超过 70,000 个主体（IAM 用户和角色合并）的组织生成内部访问调查发现。

未使用的访问调查发现

根据创建分析器时指定的天数，为选定账户或组织内的 IAM 实体（主体）生成未使用的访问调查发现。如果满足以下条件之一，则分析器下次扫描实体时会生成新的调查发现：

- 角色在指定的天数内处于非活动状态。
- 未使用的权限、未使用的用户密码或未使用的用户访问密钥超过指定的天数。

> **Note**
>
> 未使用的访问调查发现只能通过 ListFindingsV2 API 操作获得。

IAM Access Analyzer 如何生成未使用访问的调查发现

要分析未使用访问，即使您已经创建了分析器来为资源生成外部或外部访问调查发现，也必须为角色创建一个单独的未使用访问调查发现分析器。

创建未使用访问分析器后，IAM Access Analyzer 会查看访问活动来识别未使用访问。IAM Access Analyzer 会检查 Amazon 组织和账户中所有 IAM 用户、IAM 角色（包括服务角色）、用户访问密钥和用户密码的上次访问信息。这可以帮助识别未使用访问。

> **Note**
>
> 服务相关角色是一种特殊类型的服务角色，它与 Amazon Web Services 服务相关联并由该服务拥有。未使用的访问分析器不会分析服务相关角色。

对于活跃的 IAM 角色和用户，IAM Access Analyzer 使用 IAM 服务和操作的上次访问信息来识别未使用的权限。这样，您就可以在 Amazon 组织和账户级别扩展审查过程。您还可以使用上次访问的操作信息来深入调查各个角色。这样可更精细地了解哪些特定权限未被利用。

通过创建专用于未使用访问的分析器，您可以全面查看和识别 Amazon 环境中的未使用访问，从而补充现有外部访问分析器生成的调查结果。

Amazon Identity and Access Management Access Analyzer 入门

通过本主题中的信息了解使用和管理 Amazon Identity and Access Management Access Analyzer 的要求。

使用 IAM Access Analyzer 所需的权限

要成功配置和使用 IAM Access Analyzer，您使用的账户必须获得所需的权限。

IAM Access Analyzer 的 Amazon 托管策略

Amazon Identity and Access Management Access Analyzer 提供 Amazon 托管策略，可帮助您快速入门。

- IAMAccessAnalyzerFullAccess：允许管理员完全访问 IAM Access Analyzer。此策略还允许创建与服务关联的角色，以便允许 IAM Access Analyzer 分析您的账户或 Amazon 企业。
- IAMAccessAnalyzerReadOnlyAccess：允许只读访问 IAM Access Analyzer。您必须将其他策略添加到 IAM 身份（用户、用户组或角色），以允许他们查看他们的发现结果。

IAM Access Analyzer 定义的资源

要查看 IAM Access Analyzer 定义的资源，请参阅《服务授权参考》中的 IAM Access Analyzer 定义的资源类型。

所需的 IAM Access Analyzer 服务权限

IAM Access Analyzer 使用名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色（SLR）。此 SLR 授予服务只读访问权限，以使用基于资源的策略分析 Amazon 资源，并代表您分析未使用的访问。在以下情况下，服务会在您的账户中创建角色：

- 您可以创建一个外部访问分析器，将您的账户作为信任区域。
- 还可以创建一个未使用的访问分析器，将您的账户作为选定账户。
- 您可以创建一个以您的账户作为信任区域的内部访问分析器。

有关更多信息，请参阅 将服务关联角色用于 Amazon Identity and Access Management Access Analyzer。

> **Note**
>
> IAM Access Analyzer 是区域性的。对于外部和内部访问，您必须在每个区域单独启用 IAM Access Analyzer。
> 对于未使用的访问，分析器的调查发现不会因区域而变化。不需要在您拥有资源的每个区域创建分析器。

在某些情况下，在 IAM Access Analyzer 中创建分析器后，加载的调查发现页面或控制面板没有调查发现或摘要。此情况可能是因用于填充结果的控制台中出现延迟导致的。您可能需要手动刷新浏览器，或稍后返回查看您的调查发现或摘要。如果您仍然没有看到外部访问分析器的任何调查发现，这是因为您的账户中没有可供外部实体访问的受支持资源。如果将向外部实体授予访问权限的策略应用于资源，则 IAM Access Analyzer 会生成结果。

> **Note**
>
> 对于外部访问分析器，修改策略后，IAM Access Analyzer 可能需要长达 30 分钟的时间来分析资源，然后生成新的调查发现或更新现有的资源访问调查发现。
> 当您创建内部访问分析器时，可能需要几分钟或几小时才能获得调查发现。初始扫描后，IAM Access Analyzer 每 24 小时自动重新扫描所有策略。
> 对于所有类型的访问分析器，调查发现的更新可能不会立即反映在控制面板中。

查看调查发现控制面板所需的 IAM Access Analyzer 权限

要查看 IAM Access Analyzer 调查发现控制面板，必须授予您使用的账户访问权限，以执行以下所需的操作:

- `GetAnalyzer`
- `ListAnalyzers`
- `GetFindingsStatistics`

查看 IAM Access Analyzer 定义的所有操作，请参阅《服务授权参考》中的 IAM Access Analyzer 定义的操作。

IAM Access Analyzer 状态

要查看分析器的状态，请选择 Analyzers (分析器)。为组织或账户创建的分析器可具有以下状态：

Status	说明
活跃	对于外部和内部访问分析器，分析器将主动监控其信任区域内的资源。分析器会主动生成新的结果并更新现有结果。 对于未使用的访问分析器，分析器将主动监控选定组织或 Amazon Web Services 账户 在指定跟踪期限内的未使用访问。分析器会主动生成新的结果并更新现有结果。
Creating	分析器的创建过程仍在进行中。在创建过程完成后，分析器将变为活动状态。
已禁用	分析器因 Amazon Organizations 管理员执行的操作而被禁用。例如，删除作为 IAM Access Analyzer 的委派管理员的分析器的账户。当分析器处于禁用状态时，不会生成新的调查发现或更新现有调查发现。

Status	说明
失败	由于配置问题，分析器创建失败。分析器将不会生成任何结果。删除该分析器并创建新的分析器。

创建 IAM Access Analyzer 外部访问分析器

要在某个区域启用外部访问分析器，必须在该区域创建一个分析器。您必须在要监控资源访问的每个区域中创建一个外部访问分析器。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。

将 Amazon Web Services 账户作为信任区域创建一个外部访问分析器

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择资源分析 - 外部访问。
5. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
6. 输入分析器的名称。
7. 选择当前账户作为分析器的信任区域。

   > **Note**
   >
   > 如果您的账户不是 Amazon Organizations 管理账户或委托管理员账户，则您只能创建一个将您的账户作为信任区域的分析器。

8. 可选。添加要应用于分析器的所有标签。
9. 选择 Create analyzer (创建分析器)。

创建外部访问分析器以启用 IAM Access Analyzer 时，系统会在账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

将组织作为信任区域创建一个外部访问分析器

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择资源分析 - 外部访问。
5. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
6. 输入分析器的名称。
7. 选择当前组织作为分析器的信任区域。
8. 可选。添加要应用于分析器的所有标签。
9. 选择提交。

创建以组织作为信任区域的外部访问分析器时，系统会在组织的每个账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

管理 IAM Access Analyzer 外部访问分析器

要在某个区域启用外部访问分析器，必须在该区域创建一个分析器。您必须在要监控资源访问的每个区域中创建一个外部访问分析器。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。

更新外部访问分析器

按照以下过程更新外部访问分析器。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 在分析器部分，选择要管理的外部访问分析器的名称。
4. 在存档规则选项卡上，您可以为分析器创建、编辑或删除存档规则。有关更多信息，请参阅 存档规则。
5. 在标签选项卡上，您可以为分析器管理和创建标签。有关更多信息，请参阅 Amazon Identity and Access Management 资源的标签。

删除外部访问分析器

按照以下过程删除外部访问分析器。删除分析器后，将不再监控资源，也不会生成新的调查发现。分析器生成的所有调查发现都将被删除。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 在分析器部分，选择要删除的外部访问分析器的名称。
4. 选择删除分析器。
5. 输入 delete 并选择删除以确认删除分析器。

创建 IAM Access Analyzer 内部访问分析器

要在某个区域启用内部访问分析器，必须在该区域创建一个分析器。您必须在要监控资源访问的每个区域中创建一个内部访问分析器。

IAM Access Analyzer 根据每个分析器每月监控的资源数量对内部访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。
> IAM Access Analyzer 无法为包含超过 70,000 个主体（IAM 用户和角色合并）的组织生成内部访问调查发现。
> 您只能在 Amazon 组织中创建一个组织级别内部访问分析器。

创建以 Amazon Web Services 账户作为信任区域的内部访问分析器

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择资源分析 - 内部访问。
5. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
6. 输入分析器的名称。
7. 选择当前账户作为分析器的信任区域。

> **Note**
>
> 如果您的账户不是 Amazon Organizations 管理账户或委托管理员账户，则您只能创建一个将您的账户作为信任区域的分析器。

8. 在要分析的资源部分，添加分析器要监控的资源。

- 要按账户添加资源，请选择添加 > 添加选定账户中的资源。
  1. 选择所有支持的资源类型，或者选择定义特定的资源类型，然后从资源类型列表中选择资源类型。

     内部访问分析器支持以下资源类型：

     - Amazon Simple Storage Service 存储桶
     - Amazon Simple Storage Service 目录存储桶
     - Amazon Relational Database Service 数据库快照
     - Amazon Relational Database Service 数据库集群快照
     - Amazon DynamoDB Streams
     - Amazon DynamoDB 表
  2. 选择添加资源。
- 要按 Amazon 资源名称 (ARN) 添加资源，请选择添加资源 > 通过粘贴资源 ARN 来添加资源。

  > **Note**
  >
  > ARN 必须完全匹配 - 不支持通配符。对于 Amazon S3，仅支持存储桶 ARN。不支持 Amazon S3 对象 ARN 和前缀。

  1. 对于每个资源 ARN，输入账户所有者 ID 和资源 ARN（以逗号分隔）。每行输入一个账户所有者 ID 和资源 ARN。
  2. 选择添加资源。
- 要通过 CSV 文件添加资源，请选择添加资源 > 通过上传 CSV 添加资源。

  您可以使用 Amazon 资源探索器 搜索您账户中的资源并导出 CSV 文件。然后，您可以上传 CSV 文件来配置分析器要监控的资源。

  1. 选择选择文件，然后从您的计算机中选择 CSV 文件。

2. 选择添加资源。

9. 可选。添加要应用于分析器的所有标签。
10. 选择 Create analyzer (创建分析器)。

创建内部访问分析器以启用 IAM Access Analyzer 时，系统会在账户中创建名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

创建以组织作为信任区域的内部访问分析器

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择资源分析 - 内部访问。
5. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
6. 输入分析器的名称。
7. 选择整个组织作为分析器的信任区域。
8. 在要分析的资源部分，添加分析器要监控的资源。

   - 要为账户添加资源，请选择添加资源 > 添加选定账户中的资源。

     1. 选择所有支持的资源类型，或者选择定义特定的资源类型，然后从资源类型列表中选择资源类型。

        内部访问分析器支持以下资源类型：

        - Amazon Simple Storage Service 存储桶
        - Amazon Simple Storage Service 目录存储桶
        - Amazon Relational Database Service 数据库快照
        - Amazon Relational Database Service 数据库集群快照
        - Amazon DynamoDB Streams
        - Amazon DynamoDB 表

     2. 要从您的组织选择账户，请选择从组织中选择。在选择账户部分，选择层次结构以按组织结构选择账户，或选择列表以从您的组织中所有账户的列表中选择账户。

        要手动输入组织中的账户，请选择输入 Amazon 账户 ID。在 Amazon 账户 IDAmazon Web Services 账户 字段中输入一个或多个 ID（用逗号分隔）。

3. 选择添加资源。

- 要按 Amazon 资源名称 (ARN) 添加资源，请选择添加资源 > 通过粘贴资源 ARN 来添加资源。

> **Note**
>
> ARN 必须完全匹配 - 不支持通配符。对于 Amazon S3，仅支持存储桶 ARN。不支持 Amazon S3 对象 ARN 和前缀。

1. 对于每个资源 ARN，输入账户所有者 ID 和资源 ARN（以逗号分隔）。每行输入一个账户所有者 ID 和资源 ARN。
2. 选择添加资源。

- 要通过 CSV 文件添加资源，请选择添加资源 > 通过上传 CSV 添加资源。

  您可以使用 Amazon 资源探索器 搜索您账户中的资源并导出 CSV 文件。然后，您可以上传 CSV 文件来配置分析器要监控的资源。

1. 选择选择文件，然后从您的计算机中选择 CSV 文件。
2. 选择添加资源。

9. 可选。添加要应用于分析器的所有标签。
10. 选择提交。

在创建以组织作为信任区域的内部访问分析器时，系统会在组织的每个账户中创建名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

管理 IAM Access Analyzer 内部访问分析器

要在某个区域启用内部访问分析器，必须在该区域创建一个分析器。您必须在要监控资源访问的每个区域中创建一个内部访问分析器。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。

更新内部访问分析器

按照以下过程更新内部访问分析器。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 在分析器部分中，选择要管理的内部访问分析器的名称。
4. 在存档规则选项卡上，您可以为分析器创建、编辑或删除存档规则。有关更多信息，请参阅 存档规则。
5. 在标签选项卡上，您可以为分析器管理和创建标签。有关更多信息，请参阅 Amazon Identity and Access Management 资源的标签。
6. 在资源选项卡上，在要分析的资源部分中选择编辑。

   a. 要按账户添加资源，请选择添加资源 > 添加选定账户中的资源。

      1. 选择所有支持的资源类型，或者选择定义特定的资源类型，然后从资源类型列表中选择资源类型。

         内部访问分析器支持以下资源类型：

         - Amazon Simple Storage Service 存储桶
         - Amazon Simple Storage Service 目录存储桶
         - Amazon Relational Database Service 数据库快照
         - Amazon Relational Database Service 数据库集群快照
         - Amazon DynamoDB Streams
         - Amazon DynamoDB 表

      2. 选择添加资源。

   b. 要按 Amazon 资源名称 (ARN) 添加资源，请选择添加资源 > 通过粘贴资源 ARN 来添加资源。

      > **Note**
      >
      > ARN 必须完全匹配 - 不支持通配符。对于 Amazon S3，仅支持存储桶 ARN。不支持 Amazon S3 对象 ARN 和前缀。

      1. 对于每个资源 ARN，输入账户所有者 ID 和资源 ARN（以逗号分隔）。每行输入一个账户所有者 ID 和资源 ARN。
      2. 选择添加资源。

   c. 要通过 CSV 文件添加资源，请选择添加资源 > 通过上传 CSV 添加资源。

您可以使用 Amazon 资源探索器 搜索您账户中的资源并导出 CSV 文件。然后，您可以上传 CSV 文件来配置分析器要监控的资源。

1. 选择选择文件，然后从您的计算机中选择 CSV 文件。
2. 选择添加资源。

d. 要从分析器中删除资源，请选中要删除的资源旁边的复选框，然后选择删除。

e. 选择保存更改。

> **Note**
>
> 分析器的任何更新都将在 24 小时内的下一次自动重新扫描中进行评估。

删除内部访问分析器

按照以下过程删除内部访问分析器。删除分析器后，将不再监控资源，也不会生成新的调查发现。分析器生成的所有调查发现都将被删除。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 在分析器部分中，选择要删除的内部访问分析器的名称。
4. 选择删除分析器。
5. 输入 delete 并选择删除以确认删除分析器。

创建 IAM Access Analyzer 未使用的访问分析器

为当前账户创建未使用的访问分析器

使用以下过程为单个 Amazon Web Services 账户 创建未使用的访问分析器。对于未使用的访问，分析器的调查发现不会因区域而变化。不需要在您拥有资源的每个区域创建分析器。

IAM Access Analyzer 根据每个分析器每月分析的 IAM 角色和用户数量对未使用的访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择主体分析 - 未使用的访问。
5. 输入分析器的名称。
6. 对于跟踪周期，输入分析的天数。分析器将仅评估选定账户内整个跟踪周期存在的 IAM 实体的权限。例如，如果将跟踪周期设置为 90 天，则只会分析至少 90 天的权限，如果在此期间未显示任何使用情况，则会生成调查发现。您可以输入 1 至 365 天之间的值。
7. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
8. 对于选定账户，选择当前账户。

   > **Note**
   >
   > 如果您的账户不是 Amazon Organizations 管理账户或委派管理员账户，则只能创建一个将您的账户作为信任区域的分析器。

9. 可选。在排除带有标签的 IAM 用户和角色部分中，您可以为 IAM 用户和角色指定键值对，以从未使用的访问分析中排除。不会为与键值对匹配的已排除 IAM 用户和角色生成调查发现。为标签键输入长度为 1 到 128 个字符并且不以 `aws:` 为前缀的值。对于值，您可以输入长度为 0 到 256 个字符的值。如果您未输入值，则规则将应用于具有指定的标签键的所有主体。选择添加新的排除项以添加要排除的其他键值对。
10. 可选。添加要应用于分析器的所有标签。
11. 选择 Create analyzer (创建分析器)。

创建未使用的访问分析器以启用 IAM Access Analyzer 时，系统会在账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

使用当前组织创建未使用的访问分析器

使用以下过程为组织创建一个未使用的访问分析器，以集中查看组织中的所有 Amazon Web Services 账户。对于未使用的访问分析，分析器的调查发现不会因区域而变化。不需要在您拥有资源的每个区域创建分析器。

IAM Access Analyzer 根据每个分析器每月分析的 IAM 角色和用户数量对未使用的访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

> **Note**
>
> 如果成员账户从组织中删除，未使用的访问分析器将在 24 小时后停止为该账户生成新的调查发现和更新现有调查发现。与从组织中删除的成员账户关联的调查发现将在 90 天后永久删除。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 选择 Create analyzer (创建分析器)。
4. 在分析部分，选择主体分析 - 未使用的访问。
5. 输入分析器的名称。
6. 对于跟踪周期，输入分析的天数。分析器将仅评估选定组织的账户内整个跟踪周期存在的 IAM 实体的权限。例如，如果将跟踪周期设置为 90 天，则只会分析至少 90 天的权限，如果在此期间未显示任何使用情况，则会生成调查发现。您可以输入 1 至 365 天之间的值。
7. 在分析器详细信息部分，确认显示的区域是您要启用 IAM Access Analyzer 的区域。
8. 对于选定账户，选择当前组织。
9. 可选。在从分析中排除 Amazon Web Services 账户部分中，您可以在组织中选择 Amazon Web Services 账户，以从未使用的访问分析中排除。不会为排除的账户生成调查发现。

    a. 若要指定要排除的个人账户 ID，请选择指定 Amazon Web Services 账户 ID，然后在 Amazon Web Services 账户 ID 字段中输入用逗号分隔的账户 ID。选择排除。然后，这些账户将列在要排除的 Amazon Web Services 账户表中。

    b. 要从您的组织的账户列表中选择要排除的账户，请选择从组织中选择。

        i. 您可以在从组织中排除账户字段中按名称、电子邮件和账户 ID 搜索账户。

        ii. 选择层级以按组织单位查看您的账户，或选择列表以查看组织中所有个人账户的列表。

iii. 选择排除所有当前账户以排除组织单位中的所有账户，或者选择排除以排除个人账户。

然后，这些账户将列在要排除的 Amazon Web Services 账户表中。

> **Note**
>
> 排除的账户不能包含组织分析器的所有者账户。向您的组织添加新账户后，它们不会被排除在分析范围之外，即使您之前已排除某个组织单位内的所有当前账户。有关创建未使用的访问分析器后排除账户的更多信息，请参阅管理 IAM Access Analyzer 未使用的访问分析器。

10. 可选。在排除带有标签的 IAM 用户和角色部分中，您可以为 IAM 用户和角色指定键值对，以从未使用的访问分析中排除。不会为与键值对匹配的已排除 IAM 用户和角色生成调查发现。为标签键输入长度为 1 到 128 个字符并且不以 `aws:` 为前缀的值。对于值，您可以输入长度为 0 到 256 个字符的值。如果您未输入值，则规则将应用于具有指定的标签键的所有主体。选择添加新的排除项以添加要排除的其他键值对。
11. 可选。添加要应用于分析器的所有标签。
12. 选择 Create analyzer (创建分析器)。

创建未使用的访问分析器以启用 IAM Access Analyzer 时，系统会在账户中创建一个名为 `AWSServiceRoleForAccessAnalyzer` 的服务相关角色。

管理 IAM Access Analyzer 未使用的访问分析器

使用本主题中的信息了解如何更新或删除现有的未使用访问分析器。

> **Note**
>
> 创建或更新分析器后，可能需要一些时间才能获得调查发现。

更新未使用的访问分析器

按照以下过程更新未使用的访问分析器。

IAM Access Analyzer 根据每个分析器每月分析的 IAM 角色和用户数量对未使用的访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择分析器设置。
3. 在分析器部分中，选择要管理的未使用访问分析器的名称。
4. 如果分析器是为组织创建的，则在排除项选项卡上的已排除的 Amazon Web Services 账户部分中选择管理。

   a. 若要指定要排除的个人账户 ID，请选择指定 Amazon Web Services 账户 ID，然后在 Amazon Web Services 账户 ID 字段中输入用逗号分隔的账户 ID。选择排除。然后，这些账户将列在要排除的 Amazon Web Services 账户表中。

   b. 要从您的组织的账户列表中选择要排除的账户，请选择从组织中选择。

      i. 您可以在从组织中排除账户字段中按名称、电子邮件和账户 ID 搜索账户。

      ii. 选择层级以按组织单位查看您的账户，或选择列表以查看组织中所有个人账户的列表。

      iii. 选择排除所有当前账户以排除组织单位中的所有账户，或者选择排除以排除个人账户。

      然后，这些账户将列在要排除的 Amazon Web Services 账户表中。

   c. 若要移除要排除的账户，请选择要排除的 Amazon Web Services 账户表中的账户旁的移除。

   d. 选择保存更改。

   > **Note**
   >
   > - 排除的账户不能包含组织分析器的所有者账户。
   > - 向您的组织添加新账户后，它们不会被排除在分析范围之外，即使您之前已排除某个组织单位内的所有当前账户。
   > - 更新分析器的排除项后，被排除的账户列表最多可能需要两天才能更新。

5. 在排除项选项卡上，在带标签的已排除 IAM 用户和角色部分中选择管理。

   a. 您可以为 IAM 用户和角色指定键值对，以从未使用的访问分析中排除。为标签键输入长度为 1 到 128 个字符并且不以 `aws:` 为前缀的值。对于值，您可以输入长度为 0 到 256 个字符的值。如果您未输入值，则规则将应用于具有指定的标签键的所有主体。

   b. 选择添加新的排除项以添加要排除的其他键值对。

   c. 若要移除要排除的键值对，请选择键值对旁的移除。

   d. 选择保存更改。

6. 在存档规则选项卡上，您可以为分析器创建、编辑或删除存档规则。有关更多信息，请参阅 存档规则。
7. 在标签选项卡上，您可以为分析器管理和创建标签。有关更多信息，请参阅 Amazon Identity and Access Management 资源的标签。

删除未使用的访问分析器

按照以下过程删除未使用的访问分析器。删除分析器后，将不再监控资源，也不会生成新的调查发现。分析器生成的所有调查发现都将被删除。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在访问分析器下，选择未使用的访问。
3. 在访问分析器下，选择分析器设置。
4. 在分析器部分中，选择要删除的未使用访问分析器的名称。
5. 选择删除分析器。
6. 输入 delete 并选择删除以确认删除分析器。

查看 IAM Access Analyzer 的调查发现控制面板

Amazon Identity and Access Management Access Analyzer 将外部、内部和未使用访问调查发现整理到可视化摘要控制面板中。控制面板可帮助您了解大规模权限的有效使用情况，并识别需要关注的账户和 Amazon 资源。您可以使用控制面板按 Amazon 组织、客户和调查发现类型查看调查发现。

对于外部和内部访问调查发现：

- 控制面板突出显示了公共访问调查发现、外部访问调查发现和内部访问调查发现之间的差异。
- 控制面板按资源类型提供了调查发现的明细。

对于未使用访问调查发现：

- 控制面板突出显示了未使用访问调查发现最多的 Amazon Web Services 账户。
- 控制面板按类型提供了调查发现的明细。

创建任何类型的访问分析器后，IAM Access Analyzer 会自动将新的调查发现添加到相关控制面板。这使您可以识别最有安全隐患的区域并确定其优先顺序。

摘要控制面板可让您全面了解 IAM Access Analyzer 在您的 Amazon 环境中检测到的访问问题。然后，您可以深入研究各个调查发现以进行进一步调查，并采取适当的措施来解决它们。

查看外部和内部访问分析器的摘要仪表板

> **Note**
>
> 创建或更新分析器后，摘要控制面板可能需要一些时间才能反映出调查发现的更新。

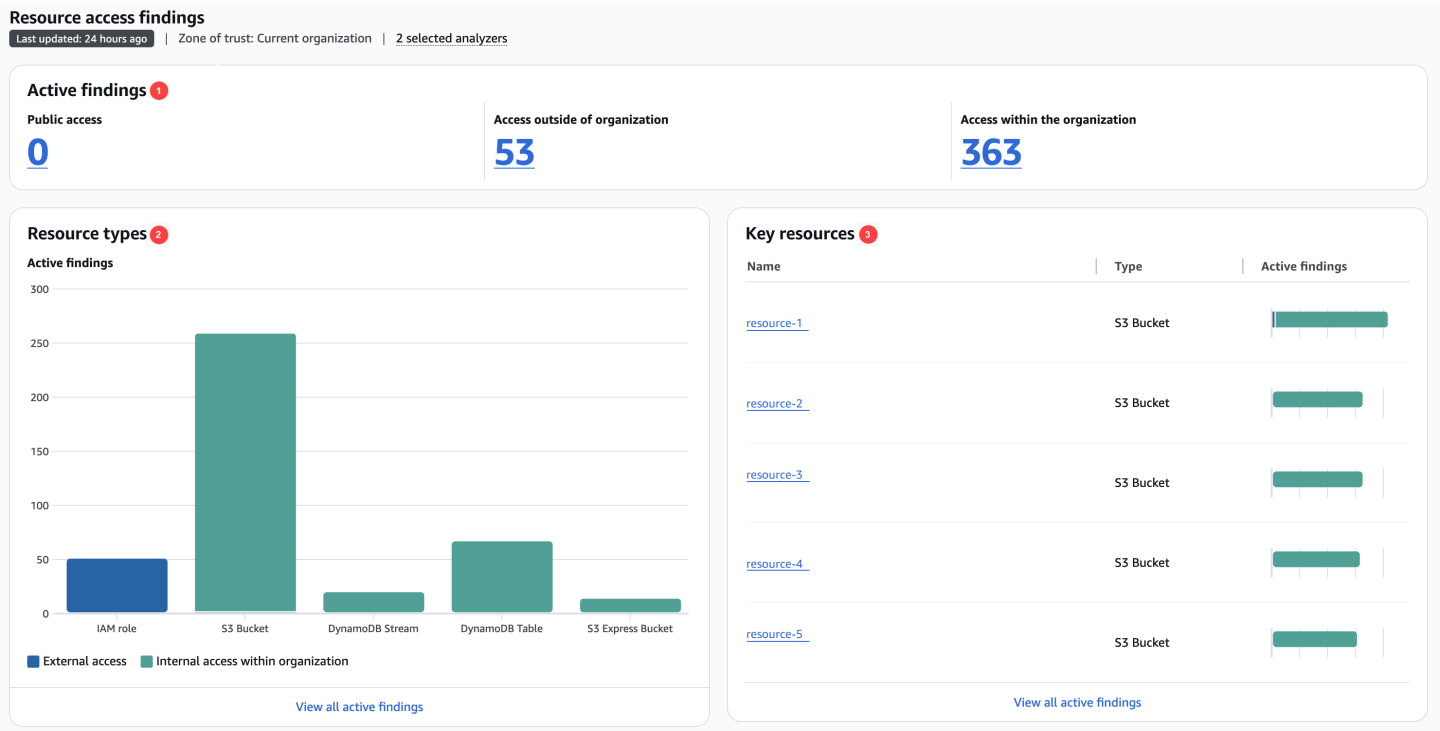
1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择访问分析器。此时将显示摘要窗口。
3. 选择选择分析器。
4. 在选择分析器窗口中，为信任区域选择组织或账户。

   > **Note**
   >
   > 只有 Amazon Organizations 管理账户或委托管理员才能选择组织作为信任区域。

5. 从资源访问分析器下拉列表中选择外部和内部访问分析器。

   > **Note**
   >
   > 您可以选择最多一个外部访问分析器和最多一个内部访问分析器。

6. 选择更新。所选外部和内部访问分析器的调查发现摘要显示在资源访问调查发现部分。



在上图中，资源调查发现控制面板在摘要页面中可见。

1. 活跃调查发现部分包括可供公共访问的活跃调查发现数量、可供账户或组织外部访问的活跃调查发现数量以及所选分析器的活跃内部访问调查发现数量。选择一个数字，列出每种类型的所有活动调查发现。
2. 资源类型部分包括所选分析器含活跃调查发现的资源类型的明细。选择查看所有活跃调查发现以获取所选分析器的活跃调查发现的完整列表。
3. 关键资源部分包括含活跃调查发现的关键资源的摘要。此信息可帮助确定对关键业务资源的调查发现的优先顺序。选择查看所有活跃调查发现以获取所选分析器的活跃调查发现的完整列表。

查看未使用访问分析器的摘要控制面板

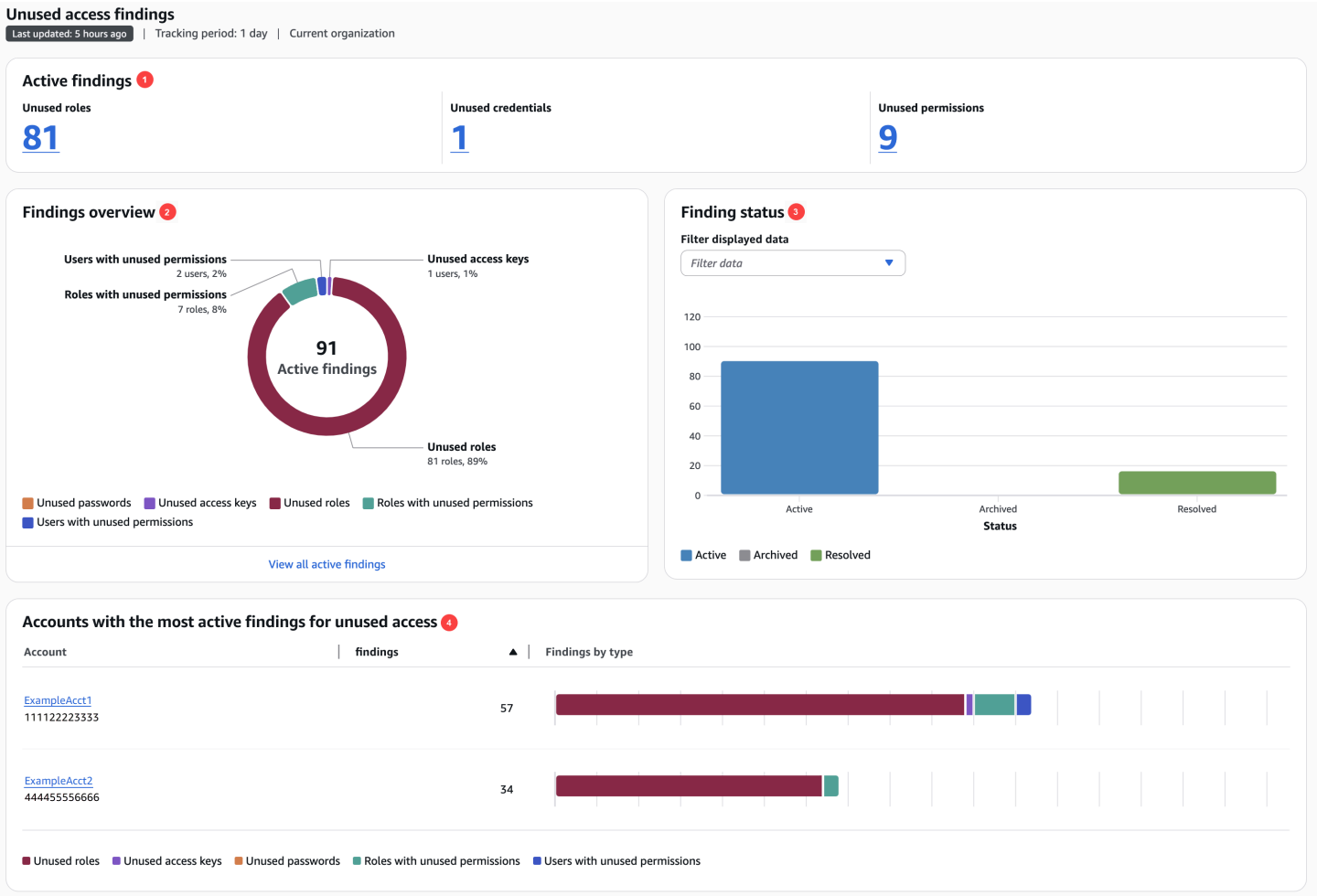
> **Note**
>
> 创建或更新分析器后，根据用户和角色的数量，摘要控制面板可能需要一些时间才能反映出调查发现的更新。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择访问分析器。将显示访问分析器摘要窗口。

3. 选择选择分析器。
4. 在选择分析器窗口中，为信任区域选择组织或账户。

> **Note**
>
> 只有 Amazon Organizations 管理账户或委托管理员才能选择组织作为信任区域。

5. 从未使用的访问分析器下拉列表中选择一个未使用的访问分析器。
6. 选择更新摘要。所选未使用访问分析器的调查发现摘要显示在未使用的访问调查发现部分。



在上图中，未使用访问调查发现控制面板在摘要页面中可见。

1. 活动调查发现部分包括您的账户或组织中未使用的角色、未使用的凭证和未使用的权限的活动调查发现数量。未使用的凭证包括未使用的访问密钥和未使用的密码调查发现。未使用的权限包括具有未使用权限的用户和角色。选择一个数字，列出每种类型的所有活动调查发现。

2. 调查发现概述部分包括活动调查发现类型的明细。选择查看所有活动调查发现，以获取分析器账户或组织的活动调查发现完整列表。
3. 调查发现状态部分包括您的账户或组织的调查发现状态（活动、已存档和已解决）的明细。您可以选择要在筛选显示的数据下拉列表中显示的调查发现状态。
4. 仅当未使用的访问分析器的选定帐户处于组织级别时，才会显示未使用的访问调查发现最多的账户部分。其中包括组织中活动调查发现最多的账户明细。这并不是组织中每个账户的详尽列表。您的分析器可能有未在该部分中列出的其他账户的活动调查发现。

查看 IAM Access Analyzer 调查发现

启用 IAM Access Analyzer 后，下一步是查看所有结果以确定结果中标识的访问是有意的还是无意的。您还可以查看调查发现，以确定针对预期访问的类似调查发现，然后创建存档规则以自动存档这些调查发现。此外，您还可以查看已存档和已解决的结果。

您应查看账户中的所有调查发现，以确定外部、内部或未使用访问是否是预期的且已批准。如果调查发现中识别的访问是预期的，则可以存档该调查发现。将调查发现存档后，其状态将变为已存档，并且该调查发现将从活动调查发现列表中删除。该结果不会被删除。您随时可以查看已存档的结果。处理您账户中的所有结果，直到不存在活动结果。当调查发现为零时，您就会知道任何新生成的活动调查发现是因为环境近期发生了变化。

查看所有类型的访问分析器的活跃调查发现

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择 Access analyzer (访问分析器)。将显示调查发现控制面板。
3. 选择选择分析器。
4. 在选择分析器窗口中，从资源访问分析器下拉列表中选择最多一个外部访问分析器和一个内部访问分析器。从未使用的访问分析器下拉列表中选择一个未使用的访问分析器。
5. 选择更新摘要。控制面板上会显示所选访问分析器的活跃调查发现摘要。在资源访问调查发现或未使用的访问调查发现部分中选择一种调查发现类型，以查看所选类型的所有活跃调查发现。

   有关查看调查发现控制面板的更多信息，请参阅 查看 IAM Access Analyzer 的调查发现控制面板。

> **Note**
>
> 仅在您有权查看分析器的结果时，才会显示结果。

外部和内部访问调查发现

> **Note**
>
> IAM Access Analyzer 根据每个区域每月监控的资源数量对内部访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

1. 在访问分析器下，选择资源分析。
2. 选择选择分析器。
3. 在选择分析器窗口中，从资源访问分析器下拉列表中选择最多一个外部访问分析器和一个内部访问分析器。
4. 选择更新摘要。

资源分析页面显示有关资源的以下详细信息以及所选访问分析器的活跃调查发现：

名称

含活跃调查发现的资源的名称。

Type

资源的类型。

所有者账户

只有当您使用某个组织作为一个或多个选定分析器的信任区域时，才会显示此列。组织中拥有结果中报告的资源的账户。

活跃调查发现

资源的活跃调查发现的数量和类型的直观表示。将鼠标悬停在该字段上可显示有关资源调查发现的更多信息。

公有访问权限

表明资源的任何调查发现是否允许公共访问。

未使用的访问调查发现

> **Note**
>
> IAM Access Analyzer 根据每月分析的 IAM 角色和用户数量对未使用的访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

1. 在访问分析器下，选择未使用的访问。
2. 选择选择分析器。
3. 在选择分析器窗口中，从未使用的访问分析器下拉列表中选择一个未使用的访问分析器。
4. 选择更新摘要。

未使用的访问页面显示有关为所选访问分析器生成调查发现的 IAM 实体的以下详细信息：

调查发现 ID

分配给结果的唯一 ID。选择调查发现 ID 以显示有关生成调查发现的 IAM 实体的其他详细信息。

调查发现类型

未使用的访问调查发现类型：未使用的访问密钥、未使用的密码、未使用的权限或未使用的角色。

IAM 实体

调查发现中报告的 IAM 实体。这可以是 IAM 用户或角色。

Amazon Web Services 账户 ID

仅当您为组织中的所有 Amazon Web Services 账户 设置分析器时，才会显示此列。拥有调查发现中报告的 IAM 实体的组织中的 Amazon Web Services 账户。

上次更新

调查发现中报告的 IAM 实体上次更新的时间，或者如果未进行任何更新，则为创建该实体的时间。

状态

调查发现的状态（活动、已存档或已解决）。

筛选 IAM Access Analyzer 调查发现

调查发现页面的默认筛选是显示所有活跃调查发现。要查看所有调查发现，请从状态下拉列表中选择全部。要查看已存档的调查发现，请选择已存档。要查看已解决的调查发现，请选择已解决。首次开始使用 IAM Access Analyzer 时，没有已存档的结果。

使用筛选条件仅显示符合指定属性条件的调查发现。要创建筛选条件，请选择要筛选的属性，然后选择该属性是否等于或包含一个值，然后输入或选择要筛选的属性值。

有关可用于创建或更新存档规则的筛选条件键的列表，请参阅 IAM Access Analyzer 筛选条件键。

筛选含活跃调查发现的资源

您可以按资源查看和筛选最多一个外部访问分析器和最多一个内部访问分析器的活跃调查发现。

筛选含活跃调查发现的资源

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择资源分析。
3. 要按资源名称进行筛选，请在搜索框中键入全部或部分资源名称。
4. 在筛选访问类型下拉列表中，选择访问类型：
   - 所有类型 – 显示含所有类型调查发现的资源。
   - 公共访问 – 仅显示含公共访问调查发现的资源。
   - 外部访问 – 仅显示含外部访问调查发现的资源。
   - 组织的内部访问 – 仅显示含内部访问调查发现的资源。
5. 在筛选资源类型下拉列表中，选择一种资源类型以仅显示选定类型的资源。

筛选外部访问调查发现

要筛选外部访问调查发现

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择分析器设置，然后在分析器部分中选择外部访问分析器。
3. 选择查看调查发现。
4. 选择搜索框以显示可用属性列表。
5. 请选择要用于筛选所显示结果的属性。

6. 为该属性选择要匹配的值。仅显示具有该结果中的值的结果。

   例如，选择资源作为属性，然后选择 Resource:，键入存储桶的部分或全部名称，并按 Enter。仅显示与筛选条件匹配的存储桶的调查发现。要创建一个筛选条件，仅显示允许公开访问的资源的调查发现，请选择 Public access（公开访问）属性，然后选择 Public access = ，然后选择 Public access = true。

您可以添加其他属性来进一步筛选显示的结果。在添加其他属性时，仅显示与筛选器中的所有条件匹配的结果。无法定义筛选器来显示与一个属性或另一个属性匹配的结果。选择清除筛选条件以清除您定义的筛选条件，并显示分析器中具有指定状态的所有调查发现。

仅当您查看将组织作为信任区域的分析器的结果时，某些字段才会显示。

以下属性可用于定义外部访问的筛选条件：

- Public access（公有访问）- 要按结果筛选允许公有访问的资源，请按公有访问进行筛选，然后选择 Public access: true（公有访问: true）。
- Resource（资源）- 要按资源进行筛选，请键入资源的完整或部分名称。
- Resource Type（资源类型）- 要按资源类型进行筛选，请从显示的列表中选择类型。
- Resource Owner Account（资源所有者账户）：使用此属性按组织中拥有调查发现中报告的资源的账户进行筛选。
- 资源控制策略限制：使用此属性按 Organizations 资源控制策略（RCP）所应用的限制类型进行筛选。要了解更多信息，请参阅《Amazon Organizations 用户指南》中的 Resource control policies (RCPs)。
  - 无法评估 RCP：评估 RCP 时出现错误。
  - 不适用：没有 RCP 限制此资源或主体。其中还包括尚不支持 RCP 的资源。
  - 适用：您的组织管理员已经通过影响资源或资源类型的 RCP 设置了限制。请联系您的组织管理员，以获取更多详细信息。
- Amazon Account：使用此属性按策略语句的主体部分中被授予访问权限的 Amazon Web Services 账户 进行筛选。要按 Amazon Web Services 账户 进行筛选，请键入完整或部分 12 位 Amazon Web Services 账户 ID，或者键入有权访问当前账户中资源的外部 Amazon 用户或角色的完整或部分账户 ARN。
- Canonical User（规范用户）：要按规范用户进行筛选，请键入为 Amazon S3 存储桶定义的规范用户 ID。要了解更多信息，请参阅 Amazon 账户标识符。
- Federated User（联合身份用户）- 要按联合身份用户进行筛选，请键入联合身份身份的完整或部分 ARN。要了解更多信息，请参阅身份提供商和联合身份验证。

- Finding ID（调查发现 ID）：要按调查发现 ID 进行筛选，请键入完整或部分调查发现 ID。
- 错误：要按错误类型进行筛选，请选择拒绝访问或内部错误。
- Principal ARN（主体 ARN）- 使用此属性可筛选 aws:PrincipalArn 条件键中使用的主体（IAM 用户、角色或组）的 ARN。要按主体 ARN 进行筛选，请键入调查发现中报告的外部 Amazon Web Services 账户 的 IAM 用户、角色或组的完整或部分 ARN。
- Principal OrgID（主体 OrgID）- 要按主体 OrgID 进行筛选，请键入与属于 Amazon 企业（指定为结果中的条件）的外部主体关联的企业 ID 的完整或部分内容。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 主体组织路径 – 要按主体组织路径进行筛选，请键入 Amazon 组织或组织单位（OU）的完整或部分 ID，该 ID 允许访问属于指定为策略中的条件的组织或 OU 的账户成员的所有外部主体。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 源账户 – 要按源账户进行筛选，请键入与资源关联的完整或部分 Amazon Web Services 账户 ID，如同 Amazon 中的某些跨服务权限中使用的那样。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- Source ARN（源 ARN）- 要按源 ARN 进行筛选，请键入指定为结果中的条件的完整或部分 ARN。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- Source IP（源 IP）- 要按源 IP 进行筛选，请键入允许外部实体在使用指定 IP 地址时访问当前账户中的资源的完整或部分 IP 地址。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 源组织 ID – 要按源组织 ID 进行筛选，请键入与资源关联的完整或部分组织 ID，如同 Amazon 中的某些跨服务权限中使用的那样。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 源组织路径 – 要按源组织路径进行筛选，请键入与资源关联的完整或部分组织单位（OU），如同 Amazon 中的某些跨服务权限中使用的那样。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- Source VPC（源 VPC）- 要按源 VPC 进行筛选，请键入允许外部实体在使用指定 VPC 时访问当前账户中的资源的全部或部分 VPC ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 源 VPC ARN：要按源 VPC ARN 进行筛选，请键入允许外部实体在使用指定 VPC 时访问当前账户中资源的全部或部分 VPC ARN。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- 源 VPCE – 要按源 VPCE 进行筛选，请键入完整或部分 VPC 端点 ID，此 ID 允许外部实体在使用指定的 VPC 端点时访问当前账户中的资源。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- VPCE 账户：要按照 VPCE 账户进行筛选，请键入拥有 VPC 端点外部实体并允许外部实体访问资源的整个或部分 12 位 Amazon Web Services 账户 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- VPCE 组织 ID — 要按照 VPCE 组织 ID 进行筛选，请键入拥有 VPC 端点外部实体并允许外部实体访问资源的整个或部分组织 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。

- VPCE 组织路径 — 要按照 VPCE 组织路径进行筛选，请键入拥有 VPC 端点外部实体并允许外部实体访问资源的整个或部分的组织单元 (OU)。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- User ID（用户 ID）：要按用户 ID 进行筛选，请键入外部 Amazon Web Services 账户 中的完整或部分 IAM 用户的用户 ID，该用户有权访问当前账户中的资源。要了解更多信息，请参阅 Amazon 全局条件上下文键。
- KMS Key ID（KMS 密钥 ID）：要按 KMS 密钥 ID 进行筛选，请键入完整或部分 KMS 密钥的密钥 ID，该密钥被指定为您当前账户中 Amazon KMS 加密的 Amazon S3 对象访问的条件。
- 会话模式 – 要按会话模式筛选 Amazon S3 目录存储桶（`ReadOnly` 或 `ReadWrite`），请键入全部或部分会话模式。要了解更多信息，请参阅《Amazon Simple Storage Service API 参考》中的 CreateSession。
- Google Audience（Google 受众）- 要按 Google 受众进行筛选，请键入指定为当前账户中的 IAM 角色访问条件的 Google 应用程序 ID 的完整或部分内容。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键。
- Cognito Audience（Cognito 受众）：要按 Amazon Cognito 受众进行筛选，请键入完整或部分 Amazon Cognito 身份池 ID，该身份池被指定为当前账户中 IAM 角色访问的条件。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键。
- Caller Account（调用方账户）：拥有或包含调用实体的账户的 Amazon Web Services 账户 ID，例如 IAM 角色、用户或账户根用户。这将由调用 Amazon KMS 的服务使用。要按调用方账户进行筛选，请键入完整或部分 Amazon Web Services 账户 ID。
- Facebook App ID（Facebook 应用程序 ID）- 要按 Facebook 应用程序 ID 进行筛选，请键入完整或部分 Facebook 应用程序 ID（或站点 ID），此 ID 指定为条件以允许使用 Login with Facebook 联合身份验证访问您当前账户中的 IAM 角色。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键中的 id 部分。
- Amazon App ID（Amazon 应用程序 ID）- 要按 Amazon 应用程序 ID 进行筛选，请键入完整或部分 Amazon 应用程序 ID（或站点 ID），此 ID 指定为条件以允许使用“以 Amazon 登录”联合身份验证访问您当前账户中的 IAM 角色。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键中的 id 部分。
- Lambda Event Source Token（Lambda 事件源令牌）- 要按随 Alexa 集成传入的 Lambda 事件源令牌进行筛选，请键入完整或部分令牌字符串。

筛选内部访问调查发现

筛选内部访问调查发现

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。

2. 选择分析器设置，然后在分析器部分中选择内部访问分析器。
3. 选择查看调查发现。
4. 选择搜索框以显示可用属性列表。
5. 请选择要用于筛选所显示结果的属性。
6. 为该属性选择要匹配的值。仅显示具有该结果中的值的结果。

   例如，选择资源作为属性，然后选择 Resource:，键入存储桶的部分或全部名称，并按 Enter。仅显示与筛选条件匹配的存储桶的调查发现。

您可以添加其他属性来进一步筛选显示的结果。在添加其他属性时，仅显示与筛选器中的所有条件匹配的结果。无法定义筛选器来显示与一个属性或另一个属性匹配的结果。选择清除筛选条件以清除您定义的筛选条件，并显示分析器中具有指定状态的所有调查发现。

仅当您查看将组织作为信任区域的分析器的结果时，某些字段才会显示。

只有在查看监控内部访问的分析器的调查发现时，才会显示下列字段：

- Resource（资源）- 要按资源进行筛选，请键入资源的完整或部分名称。
- Resource Type（资源类型）- 要按资源类型进行筛选，请从显示的列表中选择类型。
- Resource Owner Account（资源所有者账户）：使用此属性按组织中拥有调查发现中报告的资源的账户进行筛选。
- 调查发现 ID – 要按调查发现 ID 进行筛选，请键入完整或部分调查发现 ID。

筛选未使用的访问调查发现

要筛选未使用的访问调查发现

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择未使用的访问，然后在视图分析器下拉列表中选择分析器。
3. 选择搜索框以显示可用属性列表。
4. 请选择要用于筛选所显示结果的属性。
5. 为该属性选择要匹配的值。仅显示具有该结果中的值的结果。

   例如，选择调查发现类型作为属性，然后选择调查发现类型 =，再选择调查发现类型 = 未使用的角色。仅显示类型为未使用的角色的调查发现。

您可以添加其他属性来进一步筛选显示的结果。在添加其他属性时，仅显示与筛选器中的所有条件匹配的结果。无法定义筛选器来显示与一个属性或另一个属性匹配的结果。选择清除筛选条件以清除您定义的筛选条件，并显示分析器中具有指定状态的所有调查发现。

只有在查看监控未使用访问的分析器的调查发现时，才会显示下列字段：

- 调查发现类型 – 要按调查发现类型进行筛选，请按调查发现类型进行筛选，然后选择调查发现类型。
- Resource（资源）- 要按资源进行筛选，请键入资源的完整或部分名称。
- Resource Type（资源类型）- 要按资源类型进行筛选，请从显示的列表中选择类型。
- Resource Owner Account（资源所有者账户）：使用此属性按组织中拥有调查发现中报告的资源的账户进行筛选。
- 调查发现 ID – 要按调查发现 ID 进行筛选，请键入完整或部分调查发现 ID。

存档 IAM Access Analyzer 调查发现

在获得有意访问资源的调查发现时，您可以将调查发现存档。例如，针对已批准的工作流访问的 Amazon S3 存储桶的外部或内部访问调查发现，或可能仍然需要的访问密钥的未使用访问调查发现。存档调查发现时，它将从活动调查发现列表中清除。存档的结果不会被删除。您可以筛选调查发现页面以显示存档的调查发现，并可随时取消存档。

从 Findings (结果) 页对结果进行存档

1. 选中要存档的一个或多个调查发现旁边的复选框。
2. 选择操作，然后选择存档。

   屏幕顶部将显示一条确认消息。

要从调查发现详细信息页面存档调查发现

1. 为要存档的结果选择 Finding ID (结果 ID)。
2. 选择存档。

   屏幕顶部将显示一条确认消息。

要取消对结果的存档，请重复之前的步骤，但选择 Unarchive (取消存档) 而非 Archive (存档)。在取消对结果的存档时，状态将设置为“Active (活动)”。

解决 IAM Access Analyzer 调查发现

解决资源调查发现

为了解决从意外访问生成的外部和内部访问调查发现，您应该修改策略语句以删除允许访问已识别资源的权限。

对于与 Amazon S3 存储桶相关的调查发现，请使用 Amazon S3 控制台配置该存储桶的权限。

对于 IAM 角色，请使用 IAM 控制台为列出的 IAM 角色修改信任策略。

对于其他受支持的资源，请使用控制台修改导致生成的调查发现的策略语句。

在进行更改以解决资源调查发现后，例如修改应用于 IAM 角色的策略，IAM Access Analyzer 将再次扫描资源。如果删除了对资源的访问，则调查发现的状态将更改为已解决。调查发现随后将显示在已解决的调查发现列表中，而不是活动调查发现列表中。

> **Note**
>
> 这不适用于错误调查发现。当 IAM Access Analyzer 无法分析某个资源时，它将生成一个错误调查发现。如果已解决导致 IAM Access Analyzer 无法分析该资源的问题，则错误调查发现将被完全移除，而不是变为已解决的调查发现。有关更多信息，请参阅 IAM Access Analyzer 错误调查发现。

如果所做的更改导致通过其他方式对资源进行外部或内部访问（例如，通过不同的主体或其他权限），则 IAM Access Analyzer 将解决原始调查发现并生成新的活动调查发现。如果您所做的更改导致内部错误或“访问被拒绝”错误，则所有与资源特定访问相关的活动“非错误”调查发现都将得到解决，并生成新的错误调查发现。

> **Note**
>
> 对于外部访问分析器，修改策略后，IAM Access Analyzer 可能需要最多 30 分钟才能再次分析资源，然后更新调查发现。
> 对于内部访问分析器，IAM Access Analyzer 可能需要几分钟或几小时才能再次分析资源，然后更新调查发现。IAM Access Analyzer 每 24 小时自动重新扫描所有策略。
> 已解决的结果将在上次更新到查找状态后 90 天被删除。

解决未使用访问调查发现

IAM Access Analyzer 会根据调查发现的类型提供解决未使用访问分析器调查发现的建议步骤。

在进行更改以解决未使用的访问调查发现后，下次运行未使用的访问分析器时，调查发现的状态将变为已解决。调查发现不再显示在活动调查发现列表中，而是显示在已解决的调查发现列表中。如果您所做的更改仅部分解决了未使用的访问调查发现，则现有调查发现将变为已解决，但会生成新的调查发现。例如，如果仅删除调查发现中部分未使用的权限，而不是全部。

IAM Access Analyzer 根据每月分析的 IAM 角色和用户数量对未使用的访问分析收费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

解决未使用的权限调查发现问题

对于未使用的权限调查发现，IAM Access Analyzer 可以建议要从 IAM 用户或角色中移除的策略，并提供新的策略来替换现有权限策略。以下情况不支持策略建议：

- 未使用的权限调查发现适用于用户组中的 IAM 用户。
- 未使用的权限调查发现适用于 IAM Identity Center 的 IAM 角色。
- 未使用的权限调查发现具有包含 `notAction` 元素的现有权限策略。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择未使用的访问。
3. 选择调查发现类型为未使用的权限的调查发现。
4. 在建议部分，如果建议的策略列中列出了策略，请选择预览策略以查看现有策略，并使用建议的策略来替换现有策略。如果有多个建议的策略，则可以选择下一个策略和上一个策略来查看每个现有策略和建议的策略。
5. 选择下载 JSON下载一个 .zip 文件，其中包含所有建议策略的 JSON 文件。
6. 创建建议的策略并将其附加到 IAM 用户或角色。有关更多信息，请参阅更改用户的权限（控制台）和修改角色权限策略（控制台）。
7. 从 IAM 用户或角色中移除现有权限策略列中列出的策略。有关更多信息，请参阅从用户删除权限（控制台）和修改角色权限策略（控制台）。

解决未使用的角色调查发现问题

对于未使用的角色调查发现，IAM Access Analyzer 建议删除未使用的 IAM 角色。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择未使用的访问。
3. 选择调查发现类型为未使用的角色的调查发现。
4. 在建议部分，查看 IAM 角色的详细信息。
5. 删除 IAM 角色。有关更多信息，请参阅删除 IAM 角色（控制台）。

解决未使用的访问密钥调查发现问题

对于未使用的访问密钥调查发现，IAM Access Analyzer 建议停用或删除未使用的访问密钥。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择未使用的访问。
3. 选择调查发现类型为未使用的访问密钥的调查发现。
4. 在建议部分，查看访问密钥的详细信息。
5. 停用或删除访问密钥。有关更多信息，请参阅管理访问密钥（控制台）。

解决未使用的密码调查发现问题

对于未使用的密码调查发现，IAM Access Analyzer 建议删除 IAM 用户未使用的密码。

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择未使用的访问。
3. 选择调查发现类型为未使用的密码的调查发现。
4. 在建议部分，查看 IAM 用户的详细信息。
5. 删除 IAM 用户的密码。有关更多信息，请参阅创建、更改或删除 IAM 用户密码（控制台）。

IAM Access Analyzer 错误调查发现

当 IAM Access Analyzer 分析资源时，它通常会生成显示谁有权访问您的资源的调查发现。然而，在某些情况下，分析器可能会遇到阻止其完成分析的问题。在这些情况下，IAM Access Analyzer 会生成错误调查发现。

错误调查发现表明 IAM Access Analyzer 无法完成对特定资源或特定主体资源对的分析。这些调查发现可帮助您识别可能需要关注的资源，以确保进行正确的分析。

外部访问错误调查发现

识别您的账户或组织外部共享的资源的外部访问分析器可生成两种类型的错误调查发现：

- INTERNAL_ERROR – 表示 IAM Access Analyzer 在分析资源时遇到内部问题。这可能是由于服务限制或临时问题造成的。

```
{
 "findingDetails": [
  {
   "externalAccessDetails": {}
  }
 ],
 "resource": "arn:aws:iam::941407043048:role/TestAccessAnalyzer",
 "status": "ACTIVE",
 "error": "INTERNAL_ERROR",
 "createdAt": "2022-07-14T01:31:43.085000+00:00",
 "resourceType": "AWS::IAM::Role",
 "findingType": "ExternalAccess",
 "resourceOwnerAccount": "941407043048",
 "analyzedAt": "2025-03-19T06:51:46.109000+00:00",
 "id": "4b035c7d-b7d2-40e4-a6c3-9887d1a995df",
 "updatedAt": "2022-07-14T01:31:43.085000+00:00"
}
```

- ACCESS_DENIED – 表示 IAM Access Analyzer 没有分析资源所需的权限。这通常发生在 IAM Access Analyzer 的服务相关角色 (SLR) 被拒绝访问资源时。

```
{
 "findingDetails": [
  {
   "externalAccessDetails": {}
  }
 ],
 "resource": "arn:aws:kms:us-west-2:941407043048:key/01cae123-
b7f2-4488-9a05-0070a072ea2c",
 "status": "ACTIVE",
 "error": "ACCESS_DENIED",
 "createdAt": "2022-07-14T01:31:43.104000+00:00",
 "resourceType": "AWS::KMS::Key",
 "findingType": "ExternalAccess",
 "resourceOwnerAccount": "941407043048",
 "analyzedAt": "2025-03-19T06:51:46.090000+00:00",
```

```
 "id": "7ef6f04a-9d2c-4038-9cc0-2a5f00a4d8f8",
 "updatedAt": "2022-07-14T01:31:43.104000+00:00"
}
```

内部访问错误调查发现

识别您的账户或组织内的访问的内部访问分析器可生成四种类型的错误调查发现：

- PRINCIPAL_LIMIT_EXCEEDED – 当超过 3,000 个主体有权访问关键资源时生成。此错误可帮助您识别可能需要限制的访问范围过广的资源。

  如果您对环境中的资源或主体进行更改，使主体数量低于限制，则分析器将在下一次扫描期间生成正常调查发现，并且错误调查发现将被标记为已解决。

```
{
 "id": "efec28fe-b304-412f-af0f-704d0d70c79c",
 "status": "ACTIVE",
 "error": "PRINCIPAL_LIMIT_EXCEEDED",
 "resource": "arn:aws:s3:::critical-data",
 "resourceType": "AWS::S3::Bucket",
 "resourceOwnerAccount": "111122223333",
 "createdAt": "2023-11-30T00:56:56.437000+00:00",
 "analyzedAt": "2024-03-06T04:11:54.406000+00:00",
 "updatedAt": "2023-11-30T00:56:56.437000+00:00",
 "findingType": "InternalAccess",
 "findingDetails": [
  {
   "internalAccessDetails": {}
  }
 ]
}
```

- 资源级别错误（INTERNAL_ERROR 或 ACCESS_DENIED）– 与外部访问错误类似，这些错误表明分析器由于内部问题或权限问题而无法分析特定资源。当发生资源级别错误时，分析器会为该资源生成单个错误调查发现，而不是正常调查发现。

```
{
 "id": "efec28fe-b304-412f-af0f-704d0d70c79c",
 "status": "ACTIVE",
 "error": "INTERNAL_ERROR", // can be INTERNAL_ERROR or ACCESS_DENIED
 "resource": "arn:aws:s3:::critical-data",
```

```
 "resourceType": "AWS::S3::Bucket",
 "resourceOwnerAccount": "111122223333",
 "createdAt": "2023-11-30T00:56:56.437000+00:00",
 "analyzedAt": "2024-03-06T04:11:54.406000+00:00",
 "updatedAt": "2023-11-30T00:56:56.437000+00:00",
 "findingType": "InternalAccess",
 "findingDetails": [
  {
   "internalAccessDetails": {}
  }
 ]
}
```

- 主体级别错误（INTERNAL_ERROR 或 ACCESS_DENIED）– 表示分析器无法分析特定主体对特定资源的访问。与资源级别错误不同，资源可以同时对某些主体有正常调查发现，而对其他主体有错误调查发现。

```
{
 "id": "efec28fe-b304-412f-af0f-704d0d70c79c",
 "status": "ACTIVE",
 "error": "INTERNAL_ERROR", // can be INTERNAL_ERROR or ACCESS_DENIED
 "resource": "arn:aws:s3:::critical-data",
 "resourceType": "AWS::S3::Bucket",
 "resourceOwnerAccount": "111122223333",
 "createdAt": "2023-11-30T00:56:56.437000+00:00",
 "analyzedAt": "2024-03-06T04:11:54.406000+00:00",
 "updatedAt": "2023-11-30T00:56:56.437000+00:00",
 "findingType": "InternalAccess",
 "findingDetails": [
  {
   "internalAccessDetails": {
    "principal": {
     "AWS": "arn:aws:iam::111122223333:role/MyRole_1"
    },
    "principalOwnerAccount": "111122223333",
    "principalType": "IAM_ROLE",
    "accessType": "INTRA_ACCOUNT"
   }
  }
 ]
}
```

- PRINCIPAL_ERRORS_LIMIT_EXCEEDED – 当单个资源的主体级别错误调查发现过多时生成。这是一个资源级别错误调查发现，可能与同一资源的正常调查发现同时出现。


```
{
 "id": "efec28fe-b304-412f-af0f-704d0d70c79c",
 "status": "ACTIVE",
 "error": "PRINCIPAL_ERRORS_LIMIT_EXCEEDED",
 "resource": "arn:aws:s3:::critical-data",
 "resourceType": "AWS::S3::Bucket",
 "resourceOwnerAccount": "111122223333",
 "createdAt": "2023-11-30T00:56:56.437000+00:00",
 "analyzedAt": "2024-03-06T04:11:54.406000+00:00",
 "updatedAt": "2023-11-30T00:56:56.437000+00:00",
 "findingType": "InternalAccess",
 "resourceControlPolicyRestriction": "NOT_APPLICABLE",
 "serviceControlPolicyRestriction": "NOT_APPLICABLE",
 "findingDetails": [
  {
   "internalAccessDetails": {}
  }
 ]
}
```


解决错误调查发现

如果已解决导致 IAM Access Analyzer 无法分析该资源的问题，则错误调查发现将被完全移除，而不是变为已解决的调查发现。

要解决错误调查发现，请根据错误类型考虑以下方法：

- 对于 ACCESS_DENIED 错误，请验证 IAM Access Analyzer 服务相关角色是否具有访问资源所需的权限。
- 对于 PRINCIPAL_LIMIT_EXCHERED 错误，请查看资源的访问策略，并考虑将访问限制为更少的主体。
- 对于 INTERNAL_ERROR 调查发现，您可能需要等待后续分析周期，或者如果问题仍然存在，请联系 Amazon 支持人员。
- 对于 PRINCIPAL_ERRORS_LIMIT_EXCEEDED，请查看并尽可能简化受影响资源的访问模式。

在进行更改以解决根本问题后，IAM Access Analyzer 将在下一个扫描周期内再次尝试分析资源。

IAM Access Analyzer 支持的外部和内部访问资源类型

对于外部和内部访问分析器，IAM Access Analyzer 将分析应用于您启用了 IAM Access Analyzer 的区域中 Amazon 资源的基于资源的策略。它仅分析基于资源的策略。有关 IAM Access Analyzer 如何为每种资源类型生成调查发现的详细信息，请查看资源类型信息。

> **Note**
>
> 列出的受支持资源类型适用于外部和内部访问分析器。内部访问分析器并不支持外部访问分析器支持的所有资源类型。未使用的访问分析器仅支持 IAM 用户和角色。有关更多信息，请参阅了解 IAM Access Analyzer 调查发现的工作原理。

支持的外部访问资源类型

- Amazon Simple Storage Service 存储桶
- Amazon Simple Storage Service 目录存储桶
- Amazon Identity and Access Management 角色
- Amazon Key Management Service 密钥
- Amazon Lambda 函数和层
- Amazon Simple Queue Service 队列
- Amazon Secrets Manager 密钥
- Amazon Simple Notification Service 主题
- Amazon Elastic Block Store 卷和快照
- Amazon Relational Database Service 数据库快照
- Amazon Relational Database Service 数据库集群快照
- Amazon Elastic Container Registry 存储库
- Amazon Elastic File System 文件系统
- Amazon DynamoDB Streams
- Amazon DynamoDB 表

支持的内部访问资源类型

- Amazon Simple Storage Service 存储桶

- Amazon Simple Storage Service 目录存储桶
- Amazon Relational Database Service 数据库快照
- Amazon Relational Database Service 数据库集群快照
- Amazon DynamoDB Streams
- Amazon DynamoDB 表

Amazon Simple Storage Service 存储桶

当 IAM Access Analyzer 为外部访问分析器分析 Amazon S3 存储桶时，它会在应用于存储桶的 Amazon S3 存储桶策略、访问控制列表 (ACL) 或接入点（包括多区域接入点）授予对外部实体的访问权限时生成调查发现。外部实体是可用于创建筛选器（该筛选器不在信任区域内）的委托人或其他实体。例如，如果存储桶策略向其他账户授予访问权限或允许公共访问，则 IAM Access Analyzer 会生成结果。不过，如果您在存储桶上启用阻止公共访问设置，则可以阻止账户级别或存储桶级别的访问。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 Amazon S3 存储桶时，IAM Access Analyzer 会生成调查发现。

> **Note**
>
> IAM Access Analyzer 不会分析附加到跨账户存取点的访问点策略，因为访问点及其策略位于分析器账户之外。当存储桶将访问权委托给跨账户存取点，且存储桶或账户未启用屏蔽公共访问权限时，IAM Access Analyzer 将生成一个公共结果。当您启用屏蔽公共访问权限时，公共结果将得到解析，且 IAM Access Analyzer 会为跨账户存取点生成跨账户结果。

Amazon S3 屏蔽公共访问权限设置将覆盖应用于存储桶的存储桶策略。这些设置还将覆盖应用于存储桶访问点的访问点策略。IAM Access Analyzer 会在策略发生更改时在存储桶级别分析屏蔽公共访问权限设置。但是，它仅评估账户级别的屏蔽公共访问权限设置（每 6 小时一次）。这意味着，IAM Access Analyzer 可能在最多 6 小时内无法为对存储桶的公共访问生成或解决结果。例如，如果您有允许公共访问的存储桶策略，则 IAM Access Analyzer 会为该访问生成结果。如果您随后启用屏蔽公共访问权限来阻止在账户级别对存储桶进行的所有公共访问，则 IAM Access Analyzer 在最多 6 小时内不会解析存储桶策略的结果，即使已阻止对存储桶的所有公共访问。一旦在账户级别启用了阻止公共访问后，解析跨账户存取点的公共结果也可能需要长达 6 个小时。在不更改存储桶策略的情况下更改资源控制策略（RCP）不会触发对调查发现中报告的存储桶进行重新扫描。IAM Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。

对于多区域访问点，IAM Access Analyzer 会使用已建立的策略来生成查找结果。IAM Access Analyzer 每 6 小时评估一次对多区域访问点的更改。这意味着 IAM Access Analyzer 在 6 小时内不会生成或解析查找结果，即使您在此期间创建或删除了多区域访问点，或是更新了该访问点的策略。

Amazon Simple Storage Service 目录存储桶

Amazon S3 目录桶将数据分层组织到目录中，而不是通用存储桶的扁平存储结构，后者建议用于性能关键型工作负载或应用程序。对于外部访问分析器，IAM Access Analyzer 会分析允许外部实体访问目录存储桶的目录存储桶策略（包括策略中的条件语句）。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 Amazon S3 目录存储桶时，IAM Access Analyzer 会生成调查发现。

Amazon S3 目录存储桶还支持接入点，可对通过接入点向目录存储桶发出的所有请求强制执行不同的权限和网络控制。每个接入点都可以实施一种接入点策略，该策略与附加到底层目录存储桶的存储桶策略结合使用。通过目录存储桶的接入点，可以限制对特定前缀、API 操作或虚拟私有云 (VPC) 的访问权限。

> **Note**
>
> IAM Access Analyzer 不会分析附加到跨账户存取点的访问点策略，因为访问点及其策略位于分析器账户之外。当存储桶将访问权委托给跨账户存取点，且存储桶或账户未启用屏蔽公共访问权限时，IAM Access Analyzer 将生成一个公共结果。当您启用屏蔽公共访问权限时，公共结果将得到解析，且 IAM Access Analyzer 会为跨账户存取点生成跨账户结果。

有关 Amazon S3 目录存储桶的更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的使用目录存储桶。

Amazon Identity and Access Management 角色

对于 IAM 角色，IAM Access Analyzer 将分析信任策略。在角色信任策略中，您可以定义您信任代入该角色的主体。角色信任策略是附加到 IAM 中角色所必需的基于资源的策略。IAM Access Analyzer 为信任区域内可由信任区域外的外部实体访问的角色生成查询结果。

> **Note**
>
> IAM 角色是一类全局资源。如果角色信任策略向外部实体授予访问权限，则 IAM Access Analyzer 会在每个已启用的区域中生成结果。

Amazon Key Management Service 密钥

对于 Amazon KMS keys，IAM Access Analyzer 将分析应用于密钥的密钥策略和授权。如果密钥策略或授权允许外部实体访问密钥，则 IAM Access Analyzer 会生成查询结果。例如，如果您在策略语句中使用 kms:CallerAccount 条件键以允许访问特定 Amazon 账户中的所有用户，并且您指定了当前账户（当前分析器的信任区域）以外的账户，则 IAM Access Analyzer 会生成结果。要了解有关 IAM policy 语句中的 Amazon KMS 条件键的更多信息，请参阅 Amazon KMS 条件键。

在 IAM Access Analyzer 分析 KMS 密钥时，它会读取密钥元数据，例如密钥策略和授权列表。如果密钥策略不允许 IAM Access Analyzer 角色读取密钥元数据，则会生成“Access Denied (访问被拒绝)”错误结果。例如，如果以下示例策略语句是应用于密钥的唯一策略，则会导致 IAM Access Analyzer 中出现“Access denied (访问被拒绝)”错误结果。

```
{
    "Sid": "Allow access for Key Administrators",
    "Effect": "Allow",
    "Principal": {
       "AWS": "arn:aws:iam::111122223333:role/Admin"
    },
    "Action": "kms:*",
    "Resource": "*"
}
```

由于此语句仅允许 Amazon 账户 111122223333 中名为 Admin 的角色访问密钥，因此会生成“Access Denied (访问被拒绝)”错误结果，原因是 IAM Access Analyzer 无法完全分析密钥。Findings (结果) 表中会显示一个错误结果（红色文本）。该结果看起来类似于以下内容。

```
{
    "error": "ACCESS_DENIED",
    "id": "12345678-1234-abcd-dcba-111122223333",
    "analyzedAt": "2019-09-16T14:24:33.352Z",
    "resource": "arn:aws:kms:us-
west-2:1234567890:key/1a2b3c4d-5e6f-7a8b-9c0d-1a2b3c4d5e6f7g8a",
    "resourceType": "AWS::KMS::Key",
    "status": "ACTIVE",
    "updatedAt": "2019-09-16T14:24:33.352Z"
}
```

在创建 KMS 密钥时，为访问密钥而授予的权限取决于您创建密钥的方式。如果您收到密钥资源的“Access Denied (访问被拒绝)”错误结果，请将以下策略语句应用于密钥资源以向 IAM Access Analyzer 授予对密钥的访问权限。

```
{
    "Sid": "Allow IAM Access Analyzer access to key metadata",
    "Effect": "Allow",
    "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/aws-service-role/access-
analyzer.amazonaws.com/AWSServiceRoleForAccessAnalyzer"
        },
    "Action": [
        "kms:DescribeKey",
        "kms:GetKeyPolicy",
        "kms:List*"
    ],
    "Resource": "*"
},
```

在收到 KMS 密钥资源的“Access Denied (访问被拒绝)”结果，然后通过更新密钥策略来解决此结果后，此结果的状态将更新为“Resolved (已解决)”。如果存在向外部实体授予密钥权限的策略语句或密钥授权，您可能会看到密钥资源的其他结果。

Amazon Lambda 函数和层

对于 Amazon Lambda 函数，IAM Access Analyzer 会分析策略，包括策略中的条件语句，该策略将向外部实体授予对函数的访问权限。借助 Lambda，您可以将基于独有资源的策略附加到函数、版本、别名和层。IAM Access Analyzer 根据附加到函数和层的基于资源的策略报告外部访问。IAM Access Analyzer 不会根据附加到别名和使用合格 ARN 调用的特定版本的资源型策略报告外部访问。

有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的将基于资源的策略用于 Lambda 和使用版本。

Amazon Simple Queue Service 队列

对于 Amazon SQS 队列，IAM Access Analyzer 会分析允许外部实体访问队列的策略（包括策略中的条件语句）。

Amazon Secrets Manager 密钥

对于 Amazon Secrets Manager 密钥，IAM Access Analyzer 会分析允许外部实体访问密钥的策略（包括策略中的条件语句）。

Amazon Simple Notification Service 主题

IAM Access Analyzer 分析挂载到 Amazon SNS 主题的基于资源的策略，包括允许外部访问主题的策略中的条件语句。您可以允许外部账户通过基于资源的策略执行 Amazon SNS 操作，例如订阅和发布主题。如果来自信任区域之外的账户的主体可以对某个 Amazon SNS 主题执行操作，则可以从外部访问该主题。如果您在创建 Amazon SNS 主题时于策略中选择 `Everyone`，即表示该主题可供公众访问。 `AddPermission` 是向允许外部访问的 Amazon SNS 主题添加基于资源的策略的另一种方法。

Amazon Elastic Block Store 卷和快照

Amazon Elastic Block Store 卷快照没有基于资源的策略。通过 Amazon EBS 共享权限共享快照。对于 Amazon EBS 卷快照，IAM Access Analyzer 会分析允许外部实体访问快照的访问控制列表。Amazon EBS 卷快照在加密后可以与外部账户共享。未加密的卷快照可以与外部账户共享并授予公共访问权限。共享设置位于快照的 `CreateVolumePermissions` 属性中。当客户预览 Amazon EBS 快照的外部访问权限时，他们可以指定加密密钥作为快照已加密的指示器，这类似于 IAM Access Analyzer 预览版处理 Secrets Manager 密钥的方式。

Amazon Relational Database Service 数据库快照

Amazon RDS 数据库快照没有基于资源的策略。通过 Amazon RDS 数据库权限共享数据库快照，并且只能共享手动数据库快照。对于外部访问分析器，IAM Access Analyzer 会分析允许外部实体访问 Amazon RDS DB 快照的访问控制列表。未加密的数据库快照可以公开。加密的数据库快照不能公开共享，但可以与最多 20 个其他账户共享。有关更多信息，请参阅创建数据库快照。IAM Access Analyzer 将导出数据库手动快照（例如，导出到 Amazon S3 存储桶）的能力视为可信访问。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 Amazon RDS DB 快照时，IAM Access Analyzer 会生成调查发现。

> **Note**
>
> IAM Access Analyzer 无法识别直接在数据库自身上配置的公共或跨账户访问权限。IAM Access Analyzer 仅识别在 Amazon RDS 数据库快照上配置的公共或跨账户访问的结果。

Amazon Relational Database Service 数据库集群快照

Amazon RDS 数据库集群快照没有基于资源的策略。通过 Amazon RDS 数据库集群权限共享快照。对于外部访问分析器，IAM Access Analyzer 会分析允许外部实体访问 Amazon RDS DB 集群快照的访问控制列表。未加密的集群快照可以公开。加密的集群快照无法公开共享。未加密和加密的集群快照均可

以与最多 20 个其他账户共享。有关更多信息，请参阅创建数据库集群快照。IAM Access Analyzer 将导出数据库集群快照（例如，导出到 Amazon S3 存储桶）的能力视为可信访问。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 Amazon RDS DB 集群快照时，IAM Access Analyzer 会生成调查发现。

> **Note**
>
> IAM Access Analyzer 的结果不包括监控使用 Amazon Resource Access Manager 与其他 Amazon Web Services 账户 或组织共享 Amazon RDS 数据库集群和克隆。IAM Access Analyzer 仅识别在 Amazon RDS 数据库集群快照上配置的公共或跨账户访问的结果。

Amazon Elastic Container Registry 存储库

对于 Amazon ECR 存储库，IAM Access Analyzer 分析基于资源的策略，包括策略中允许外部实体访问存储库的条件语句（类似于其他资源类型，如 Amazon SNS 主题和 Amazon EFS 文件系统）。对于 Amazon ECR 存储库，主体必须通过基于身份的策略的获得 `ecr:GetAuthorizationToken` 的权限，才能被视为外部可用。

Amazon Elastic File System 文件系统

对于 Amazon EFS 文件系统，IAM Access Analyzer 会分析允许外部实体访问文件系统的策略（包括策略中的条件语句）。如果来自信任区域之外的账户的主体可以在该文件系统上执行操作，则可以从外部访问 Amazon EFS 文件系统。访问权限由使用 IAM 的文件系统策略以及文件系统的安装方式定义。例如，将您的 Amazon EFS 文件系统安装到另一个账户视为可以从外部访问，除非该账户位于您的组织中，并且您已将该组织定义为信任区域。如果您从具有公有子网的虚拟私有云中安装文件系统，则该文件系统可以从外部访问。将 Amazon EFS 与 Amazon Transfer Family 结合使用时，如果文件系统允许公共访问，则从 Transfer Family 服务器（由与文件系统不同的账户拥有）收到的文件系统访问请求将被阻止。

Amazon DynamoDB Streams

对于外部访问分析器，如果 DynamoDB 策略允许至少一个允许外部实体访问 DynamoDB 流的跨账户操作，则 IAM Acccess Analyzer 会生成调查发现。有关 DynamoDB 支持的跨账户操作的更多信息，请参阅 Amazon DynamoDB 开发者指南中的基于资源的策略支持的 IAM 操作。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 DynamoDB 流时，IAM Access Analyzer 会生成调查发现。

Amazon DynamoDB 表

对于外部访问分析器，如果 DynamoDB 策略允许至少一个允许外部实体访问 DynamoDB 表或索引的跨账户操作，则 IAM Acccess Analyzer 会生成 DynamoDB 表的调查发现。有关 DynamoDB 支持的跨账户操作的更多信息，请参阅 Amazon DynamoDB 开发者指南中的基于资源的策略支持的 IAM 操作。

对于内部访问分析器，当您的组织或账户内的主体（用户或角色）有权访问指定的 DynamoDB 表时，IAM Access Analyzer 会生成调查发现。

IAM Access Analyzer 的委托管理员。

如果您在 Amazon Organizations 管理账户中配置 Amazon Identity and Access Management Access Analyzer，则可以在企业中添加成员账户作为委托管理员来管理企业的 IAM Access Analyzer。委派管理员有权在组织内创建和管理分析器。只有管理账户才能添加委派管理员。

IAM Access Analyzer 委派管理员是组织内的成员账户，拥有创建和管理分析器的权限，这些分析器用于分析整个组织的访问。只有管理账户才能添加、删除或更改委托管理员。

如果您添加了委派管理员，则可稍后更改为委派管理员的其他账户。执行此操作时，以前的委派管理员账户将失去对使用该账户创建的所有分析器的权限，这些分析器用于分析整个组织的访问。这些分析器将变为禁用状态，并且不再生成新的结果或更新现有结果。这些分析器的现有结果也不再可供访问。您可在以后通过将账户配置为委派管理员来再次访问它们。如果您知道您不会使用与委派管理员相同的账户，请考虑在更改委派管理员之前删除分析器。这将删除生成的所有结果。当新的委派管理员创建新分析器时，会生成相同结果的新实例。您不会丢失任何结果，只是会在其他账户中为新分析器生成结果。您可以继续使用企业管理账户（也具有管理员权限）访问企业的结果。新的委托管理员必须为 IAM Access Analyzer 创建新的分析器，以便它开始监控企业中的资源。

如果委托管理员离开 Amazon 组织，则将从账户中删除委托管理权限。账户中所有将组织作为信任区域的分析器都将变为禁用状态。这些分析器的现有结果也不再可供访问。

首次在管理账户中配置分析器时，可以在 IAM Access Analyzer 控制台的分析器设置页面上选择添加委派管理员。

> **Note**
>
> IAM Access Analyzer 根据每月每个分析器分析的 IAM 角色和用户数量对未使用的访问分析器收费。如果您在管理账户和委派管理员账户中创建了未使用的访问分析器，则要为这两个未使用的访问分析器付费。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

更改委派管理员后，新管理员必须创建分析器才能开始监控对组织中资源的访问。

添加 IAM Access Analyzer 的委托管理员

如果您在 Amazon Organizations 管理账户中配置 Amazon Identity and Access Management Access Analyzer，则可以在企业中添加成员账户作为委托管理员来管理企业的 IAM Access Analyzer。委派管理员有权在组织内创建和管理分析器。只有管理账户才能添加委派管理员。

使用控制台添加委派管理员

1. 使用企业的管理账户登录 Amazon 控制台。
2. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
3. 在访问分析器下，选择分析器设置。
4. 选择 Add delegated administrator (添加委派管理员)。
5. 在委派管理员字段中，输入组织成员账户的 Amazon Web Services 账户 号，使其成为委派管理员。

   该账户必须是您组织的成员。
6. 选择保存更改。

使用 Amazon CLI 或 Amazon 开发工具包添加委托管理员

当您使用 Amazon CLI、Amazon API（使用 Amazon SDK）或 Amazon CloudFormation 在委派管理员账户中创建分析器以分析整个组织的访问时，必须使用 Amazon Organizations API 为 IAM Access Analyzer 启用服务访问，并将成员账户注册为委派管理员。

1. 在 Amazon Organizations 中为 IAM Access Analyzer 启用受信任的服务访问。请参阅《Amazon Organizations 用户指南》中的如何启用或禁用可信访问。
2. 使用 Amazon Organizations `RegisterDelegatedAdministrator` API 操作或 `register-delegated-administrator` Amazon CLI 命令将 Amazon 企业的有效成员账户注册为委托管理员。

存档规则

存档规则会自动存档符合您在创建规则时所定义条件的新结果。还可以追溯应用存档规则，以存档符合存档规则条件的现有结果。例如，您可以创建一个存档规则，来自动存档您定期为其授予访问权限的特

定 Amazon S3 存储桶的调查发现。或者，如果您向特定主体授予对多个资源的访问权限，则可以创建一个规则，来自动存档针对向该主体授予的访问权限而生成的任何新结果。这可让您仅关注可能指示安全风险的活动结果。

在创建存档规则时，仅自动存档与规则条件匹配的新结果。不会自动存档现有结果。在创建规则时，可以在规则中为每个标识包含最多 20 个值。有关可用于创建或更新存档规则的筛选条件键的列表，请参阅 IAM Access Analyzer 筛选条件键。

> **Note**
>
> 在创建或编辑存档规则时，IAM Access Analyzer 不会验证规则筛选器中包含的值。例如，如果您添加一个规则来匹配 Amazon Web Services 账户，则 IAM Access Analyzer 将接受字段中的任何值，即使该值不是有效的 Amazon 账号。

创建存档规则

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 选择访问分析器，然后选择分析器设置。
3. 在分析器部分，选择要为其创建存档规则的分析器。
4. 在存档规则选项卡上，选择创建存档规则。
5. 如果要更改默认名称，请为规则输入一个名称。
6. 在 Rule (规则) 部分中的 Criteria (条件) 下，为该规则选择要匹配的属性。
7. 为属性值选择一个条件，例如 Contains、Is 或 Not Equals。

   可用的运算符取决于您选择的属性。
8. （可选）为属性添加其他值，或为规则添加其他条件。对于外部访问调查发现，为了确保您的规则不会存档新的调查发现以进行公开访问，您还可以包含 Public access 条件并将其设置为 false。

   要为标准添加另一个值，请选择 Add another value (添加另一个值)。要为规则添加另一个条件，请选择添加条件。
9. 添加完条件和值后，选择创建规则以仅将规则应用于新结果。选择创建和存档活动的结果，以根据规则条件存档新的和现有的结果。在结果部分中，您可以查看存档规则应用于的活动结果的列表。

例如，要为外部访问调查发现创建一个规则，以自动存档 Amazon S3 存储桶的任何调查发现：选择资源类型，然后选择 Is 作为条件。然后，从值列表中选择 S3 存储桶。

要为未使用的访问调查发现创建一个规则，以自动归档特定账户的任何调查发现：选择资源所有者账户，然后选择 Equals 作为条件。在值文本框中键入 Amazon Web Services 账户 ID。

继续定义条件以根据您的环境自定义规则，然后选择创建规则。

如果您创建一个新规则并添加多个标准，则可以通过选择 Remove this criterion（删除此标准）来从规则中删除单个标准。您可以选择 Remove value (删除值) 来删除为标准添加的值。

编辑存档规则

1. 在名称列中选择要编辑的规则名称。

   您一次只能编辑一个存档规则。
2. 为每个条件添加新条件或删除现有条件和值。
3. 选择保存更改以将规则仅应用于新结果。选择保存和存档活动的结果，以根据规则条件存档新的和现有的结果。

删除存档规则

1. 选中要删除的规则对应的复选框。
2. 选择删除。
3. 在 Delete archive rule (删除存档规则) 确认对话框中键入 **`delete`**，然后选择 Delete (删除)。

这仅会从当前区域的分析器中删除规则。您必须为在其他区域中创建的每个分析器单独删除存档规则。

使用 Amazon EventBridge 监控 Amazon Identity and Access Management Access Analyzer

通过本主题中的信息了解如何使用 Amazon EventBridge 监控 IAM Access Analyzer 结果以及如何访问预览。EventBridge 是 Amazon CloudWatch Events 的新版本。

结果事件

IAM Access Analyzer 针对每个生成的结果、对现有结果状态的更改以及删除结果的时间向 EventBridge 发送一个事件。要接收结果以及有关结果的通知，您必须在 Amazon EventBridge 中创建事件规则。在创建事件规则时，您还可以根据规则指定要触发的目标操作。例如，您可以创建一个事件规则，该规则会在从 IAM Access Analyzer 接收新结果的事件时触发 Amazon SNS 主题。有关资源控制策略（RCP）的详细信息可参见事件详细信息部分。

访问预览事件

IAM Access Analyzer 会向 EventBridge 发送事件，其中包含每个访问预览并更改为其状态。这包括首次创建访问预览（状态为“Creating”）、访问预览完成（状态为“Completed”）或访问预览创建失败（状态为“Failed”）时的事件。要接收有关访问预览的通知，您必须在 EventBridge 中创建事件规则。在创建事件规则时，您还可以根据规则指定要触发的目标操作。例如，您可以创建一个事件规则，该规则会在从 IAM Access Analyzer 接收完成的访问预览的事件时触发 Amazon SNS 主题。

事件通知频率

自您的账户中发生事件后约 1 小时内，IAM Access Analyzer 会将新结果的事件以及带状态更新的结果发送到 EventBridge。IAM Access Analyzer 还会在因保留期已过而删除已解析的结果时向 EventBridge 发送事件。对于因生成它们的分析器被删除而被删除的结果，事件将在分析器被删除后大约 24 小时发送到 EventBridge。删除结果时，不会更改其状态。相反，会将 `isDeleted` 属性设置为 `true`。IAM Access Analyzer 还会将新创建的访问预览和访问预览状态更改的事件发送到 EventBridge。

外部访问调查发现事件示例

以下是发送到 EventBridge 的 IAM Access Analyzer 外部访问调查发现事件示例。列出的 `id` 是 EventBridge 中事件的 ID。要了解更多信息，请参阅 EventBridge 中的事件和事件模式。

在 `detail` 对象中，`accountId` 和 `region` 属性的值是指结果中报告的账户和区域。`isDeleted` 属性指示事件是否来自要删除的结果。`id` 是结果 ID。`resources` 数组是一个单例，其中包含生成结果的分析器的 ARN。

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "action": [
            "s3:GetObject"
        ],
        "analyzedAt": "2019-11-21T01:22:22Z",
        "condition": {},
        "createdAt": "2019-11-20T04:58:50Z",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "isPublic": false,
        "principal": {
```

```
            "AWS": "999988887777"
        },
        "region": "us-west-2",
        "resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
}
```

IAM Access Analyzer 还会将错误结果的事件发送到 EventBridge。错误结果是在 IAM Access Analyzer 无法分析的资源时生成的结果。错误结果的事件包括一个 `error` 属性，如以下示例所示。

```
{
    "account": "111122223333",
    "detail": {
        "accountId": "111122223333",
        "analyzedAt": "2019-11-21T01:22:22Z",
        "createdAt": "2019-11-20T04:58:50Z",
        "error": "ACCESS_DENIED",
        "id": "22222222-dcba-4444-dcba-333333333333",
        "isDeleted": false,
        "region": "us-west-2",
        "resource": "arn:aws:s3:::amzn-s3-demo-bucket",
        "resourceType": "AWS::S3::Bucket",
        "status": "ACTIVE",
        "updatedAt": "2019-11-21T01:14:07Z",
        "version": "1.0"
    },
    "detail-type": "Access Analyzer Finding",
    "id": "11111111-2222-4444-aaaa-333333333333",
    "region": "us-west-2",
    "resources": [
```

```
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2019-11-21T01:22:33Z",
    "version": "0"
}
```

内部访问调查发现事件示例

以下是发送到 EventBridge 的 IAM Access Analyzer 内部访问调查发现事件示例。列出的 `id` 是 EventBridge 中事件的 ID。要了解更多信息，请参阅 EventBridge 中的事件和事件模式。

在 `detail` 对象中，`accountId` 和 `principalOwnerAccount` 属性的值是指调查发现中报告的主体的账户。`isDeleted` 属性指示事件是否来自要删除的结果。`id` 是结果 ID。`resource` 是生成调查发现的分析器的 ARN。

```
{
    "version": "0",
    "id": "b45c3678-c278-b593-6121-c155259ce1b5",
    "detail-type": "Internal Access Finding",
    "source": "aws.access-analyzer",
    "account": "111122223333",
    "time": "2025-04-08T19:42:49Z",
    "region": "us-east-1",
    "resources": [
        "arn:aws:access-analyzer:us-east-1:111122223333:analyzer/testAnalyzer"
    ],
    "detail": {
        "accessType": "INTRA_ACCOUNT",
        "action": [
            "s3:GetObject"
        ],
        "analyzedAt": "2025-04-08T03:18:43.509465073Z",
        "condition": {},
        "createdAt": "2025-04-07T21:33:49.914099224Z",
        "id": "11111111-2222-4444-aaaa-333333333333",
        "isDeleted": false,
        "findingType": "InternalAccess",
        "principal": {
            "AWS": "arn:aws:iam::111122223333:role/MyRole_6"
        },
        "principalOwnerAccount": "111122223333",
        "principalType": "IAM_ROLE",
```

```
        "resource": "arn:aws:s3:::critical-data",
        "resourceControlPolicyRestrictionType": "NOT_APPLICABLE",
        "accountId": "111122223333",
        "resourceType": "AWS::S3::Bucket",
        "serviceControlPolicyRestrictionType": "NOT_APPLICABLE",
        "status": "ACTIVE",
        "updatedAt": "2025-04-08T03:22:12.654688231Z",
        "version": "1.0"
    }
}
```

IAM Access Analyzer 还会将错误结果的事件发送到 EventBridge。错误结果是在 IAM Access Analyzer 无法分析的资源时生成的结果。错误结果的事件包括一个 `error` 属性，如以下示例所示。

```
{
    "version": "0",
    "id": "5a94b99b-e87d-a6a7-58c7-f47871532860",
    "detail-type": "Internal Access Finding",
    "source": "aws.access-analyzer-test",
    "account": "444455556666",
    "time": "2025-05-07T11:57:54Z",
    "region": "us-west-2",
    "resources": ["arn:aws:access-analyzer-beta:us-west-2:444455556666:analyzer/
example-analyzer"],
    "detail": {
        "analyzedAt": "2025-03-24T19:58:52.512329448Z",
        "createdAt": "2025-03-22T03:30:46.920200692Z",
        "id": "ef573afd-12a5-4095-87a6-bf2f25109895",
        "isDeleted": false,
        "findingType": "InternalAccess",
        "resource": "arn:aws:s3:::test-entity-88",
        "accountId": "111122223333",
        "resourceControlPolicyRestrictionType": "NOT_APPLICABLE",
        "resourceType": "AWS::S3::Bucket",
        "serviceControlPolicyRestrictionType": "NOT_APPLICABLE",
        "error": "ACCESS_DENIED", // can be INTERNAL_ERROR and ACCESS_DENIED
        "status": "ACTIVE",
        "updatedAt": "2025-03-24T20:09:39.176075014Z",
        "version": "1.0"
    }
}
```

未使用的访问调查发现相关事件示例

以下是发送到 EventBridge 的 IAM Access Analyzer 未使用的访问调查发现事件示例。列出的 `id` 是 EventBridge 中事件的 ID。要了解更多信息，请参阅 EventBridge 中的事件和事件模式。

在 `detail` 对象中，`accountId` 和 `region` 属性的值是指结果中报告的账户和区域。`isDeleted` 属性指示事件是否来自要删除的结果。`id` 是结果 ID。

```
{
    "version": "0",
    "id": "dc7ce3ee-114b-3243-e249-7f10f9054b21",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "123456789012",
    "time": "2023-09-29T17:31:40Z",
    "region": "us-west-2",
    "resources": [
       "arn:aws:access-analyzer:us-west-2:123456789012:analyzer/
integTestLongLivingAnalyzer-DO-NOT-DELETE"
       ],
    "detail": {
        "findingId": "b8ae0460-5d29-4922-b92a-ba956c986277",
        "resource": "arn:aws:iam::111122223333:role/FindingIntegTestFakeRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "111122223333",
        "createdAt": "2023-09-29T17:29:18.758Z",
        "updatedAt": "2023-09-29T17:29:18.758Z",
        "analyzedAt": "2023-09-29T17:29:18.758Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "62160bda-8e94-46d6-ac97-9670930d8ffb",
        "isDeleted": false,
        "findingType": "UnusedPermission",
        "numberOfUnusedServices": 0,
        "numberOfUnusedActions": 1
        }
    }
```

IAM Access Analyzer 还会将错误结果的事件发送到 EventBridge。错误结果是在 IAM Access Analyzer 无法分析的资源时生成的结果。错误结果的事件包括一个 `error` 属性，如以下示例所示。

```
{
```

```
    "version": "0",
    "id": "c2e7aa1a-4df7-7652-f33e-64113b8997d4",
    "detail-type": "Unused Access Finding for IAM entities",
    "source": "aws.access-analyzer",
    "account": "111122223333",
    "time": "2023-10-31T20:26:12Z",
    "region": "us-west-2",
    "resources": [
      "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/ba811f91-
de99-41a4-97c0-7481898b53f2"
      ],
    "detail": {
        "findingId": "b01a34f2-e118-46c9-aef8-0d8526b495c7",
        "resource": "arn:aws:iam::123456789012:role/TestRole",
        "resourceType": "AWS::IAM::Role",
        "accountId": "444455556666",
        "createdAt": "2023-10-31T20:26:08.647Z",
        "updatedAt": "2023-10-31T20:26:09.245Z",
        "analyzedAt": "2023-10-31T20:26:08.525Z",
        "previousStatus": "",
        "status": "ACTIVE",
        "version": "7c7a72a2-7963-4c59-ac71-f0be597010f7",
        "isDeleted": false,
        "findingType": "UnusedIAMRole",
        "error": "INTERNAL_ERROR"
        }
  }
```

访问预览事件示例

以下示例显示了在您创建访问预览时发送到 EventBridge 的第一个事件的数据。`resources` 数组是一个单例，其中包含了访问预览所关联的分析器的 ARN。在 `detail` 对象中，`id` 是指访问预览 ID，`configuredResources` 是指为其创建访问预览的资源。`status` 为 `Creating`，指访问预览状态。`previousStatus` 未指定，因为访问预览刚刚创建。

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
```

```
        "region": "us-west-2",
        "status": "CREATING",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "aaaabbbb-2222-3333-4444-555566667777",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

以下示例显示了发送到 EventBridge 以进行访问预览的事件数据，其状态从 `Creating` 更改为 `Completed`。在详细信息对象中，`id` 是指访问预览 ID。`status` 和 `previousStatus` 是指访问预览状态，其中之前的状态为 `Creating`，当前状态为 `Completed`。

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.000Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "COMPLETED",
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "11112222-3333-4444-5555-666677778888",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

以下示例显示了发送到 EventBridge 以进行访问预览的事件数据，其状态从 Creating 更改为 Failed。在 detail 对象中，id 是指访问预览 ID。status 和 previousStatus 是指访问预览状态，其中之前的状态为 Creating，当前状态为 Failed。statusReason 字段提供了指示访问预览由于资源配置无效而失败的原因代码。

```
{
    "account": "111122223333",
    "detail": {
        "accessPreviewId": "aaaabbbb-cccc-dddd-eeee-ffffaaaabbbb",
        "configuredResources": [
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ],
        "createdAt": "2020-02-20T00:00:00.00Z",
        "previousStatus": "CREATING",
        "region": "us-west-2",
        "status": "FAILED",
        "statusReason": {
            "code": "INVALID_CONFIGURATION"
        },
        "version": "1.0"
    },
    "detail-type": "Access Preview State Change",
    "id": "99998888-7777-6666-5555-444433332222",
    "region": "us-west-2",
    "resources": [
        "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/MyAnalyzer"
    ],
    "source": "aws.access-analyzer",
    "time": "2020-02-20T00:00:00.00Z",
    "version": "0"
}
```

使用控制台创建事件规则

以下过程介绍了如何使用控制台创建事件规则。

1. 访问 https://console.aws.amazon.com/events/，打开 Amazon EventBridge 控制台。
2. 使用以下值创建监控查找事件或访问预览事件的 EventBridge 规则：
   - 对于规则类型，选择具有事件模式的规则。
   - 对于 Event source（事件源），选择 Other（其他）。

- 对于 Event pattern（事件模式），选择 Custom patterns (JSON editor) [自定义模式（JSON 编辑器）]，并将以下事件模式之一粘贴到文本区域：
  - 要基于任何 IAM Access Analyzer 事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ]
}
```

  - 要基于外部访问、内部访问或未使用的访问调查发现事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Analyzer Finding",
    "Internal Access Finding",
    "Unused Access Finding for IAM entities"
  ]
}
```

  - 要仅基于外部访问调查发现事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Analyzer Finding"
  ]
}
```

  - 要仅基于内部访问调查发现事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Internal Access Finding"
```

```
  ]
}
```

- 要仅基于未使用的访问调查发现事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Unused Access Finding for IAM entities"
  ]
}
```

- 要基于访问预览事件创建规则，请使用以下模式示例：

```
{
  "source": [
    "aws.access-analyzer"
  ],
  "detail-type": [
    "Access Preview State Change"
  ]
}
```

- 对于目标类型，选择 Amazon 服务，对于选择目标，选择一个目标，例如 Amazon SNS 主题或 Amazon Lambda 函数。在收到与规则中定义的事件模式匹配的事件时将触发目标。

要详细了解关于创建规则的信息，请参阅《Amazon EventBridge 用户指南》中的创建对事件作出反应的 Amazon EventBridge 规则。

使用 CLI 创建事件规则

1. 通过以下命令为使用 Amazon CLI 的 Amazon EventBridge 创建规则。将规则名称 *TestRule* 替换为您的规则名称。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"]}"
```

2. 您可以自定义规则以便仅针对一小部分生成的结果（例如，具有特定属性的结果）触发目标操作。以下示例演示了如何创建仅针对状态为“活动”的结果触发目标操作的规则。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Analyzer Finding\"],\"detail\":{\"status\":
[\"ACTIVE\"]}}"
```

以下示例演示了如何创建仅针对状态从 `Creating` 到 `Completed` 的访问预览触发目标操作的规则。

```
aws events put-rule --name TestRule --event-pattern "{\"source\":[\"aws.access-
analyzer\"],\"detail-type\":[\"Access Preview State Change\"],\"detail\":{\"status
\":[\"COMPLETED\"]}}"
```

3. 要将 Lambda 函数定义为您所创建规则的目标，请使用以下示例命令。根据您的环境，替换 ARN 中的区域和函数名称。

```
aws events put-targets --rule TestRule --targets Id=1,Arn=arn:aws:lambda:us-
east-1:111122223333:function:MyFunction
```

4. 添加调用规则目标所需的权限。以下示例演示了如何按照前面的示例向 Lambda 函数授予权限。

```
aws lambda add-permission --function-name MyFunction --statement-id 1 --action
 'lambda:InvokeFunction' --principal events.amazonaws.com
```

将 IAM Access Analyzer 与 Amazon Security Hub CSPM 集成

Amazon Security Hub CSPM 提供跨 Amazon 的安全状态的全面视图。它可以帮助您根据安全行业标准和最佳实践评估您的环境。Security Hub CSPM 从 Amazon Web Services 账户、服务和受支持的第三方合作伙伴产品中收集安全数据。然后，它会分析您的安全趋势并确定最高优先级的安全问题。

将 IAM Access Analyzer 与 Security Hub CSPM 集成时，您可以将调查发现从 IAM Access Analyzer 发送到 Security Hub CSPM。随后，Security Hub CSPM 可以将这些调查发现纳入对您的总体安全状况的分析中。

目录

- IAM Access Analyzer 如何向 Security Hub CSPM 发送调查发现
  - IAM Access Analyzer 发送的调查发现类型

- 发送调查发现的延迟
- Security Hub CSPM 不可用时重试
- 更新 Security Hub CSPM 中的现有调查发现
- 在 Security Hub CSPM 中查看 IAM Access Analyzer 调查发现
  - 解释 Security Hub CSPM 中的 IAM Access Analyzer 调查发现名称
- IAM Access Analyzer 的典型调查发现
- 启用和配置集成
- 如何停止发送调查发现

IAM Access Analyzer 如何向 Security Hub CSPM 发送调查发现

在 Security Hub CSPM 中，安全问题按调查发现进行跟踪。一些检查结果来自其他 Amazon Web Services 服务或第三方合作伙伴检测到的问题。Security Hub CSPM 还有一套用于检测安全问题和生成调查发现的规则。

Security Hub CSPM 提供了用于管理来自所有这些来源的调查发现的工具。您可以查看和筛选调查发现列表，并查看有关每个调查发现的详细信息。有关更多信息，请参阅 Amazon Security Hub 用户指南中的查看结果。您还可以跟踪调查发现的调查状态。有关更多信息，请参阅 Amazon Security Hub 用户指南中对结果采取行动。

Security Hub CSPM 中的所有调查发现都使用名为 Amazon 安全调查发现格式（ASFF）的标准 JSON 格式。ASFF 包含有关问题根源、受影响资源以及调查发现当前状态的详细信息。有关更多信息，请参阅《Amazon Security Hub 用户指南》中的 Amazon 安全调查发现格式（ASFF）。

Amazon Identity and Access Management Access Analyzer 是一项 Amazon Web Services 服务，可将调查发现发送到 Security Hub CSPM。对于未使用的访问权限，IAM Access Analyzer 会检测授予 IAM 用户或角色的未使用访问权限并为每个用户或角色生成调查发现。然后，IAM Access Analyzer 将这些调查发现发送到 Security Hub CSPM。

对于外部访问，IAM Access Analyzer 会检测允许外部主体对组织或账户中的受支持资源进行公共访问或跨账户访问的策略语句。IAM Access Analyzer 会生成公开访问的调查发现，然后将其发送到 Security Hub CSPM。对于跨账户访问，IAM Access Analyzer 一次只能向 Security Hub CSPM 发送一个外部主体的单个调查发现。如果 IAM Access Analyzer 中有多个跨账户调查发现，则在 IAM Access Analyzer 提供下一个跨账户调查发现之前，您必须先解析单个外部主体的 Security Hub CSPM 调查发现。有关分析器信任区域之外具有跨账户存取权限的外部主体的完整列表，您必须在 IAM Access Analyzer 中查看调查发现。有关资源控制策略（RCP）的详细信息可参见资源详细信息部分。

IAM Access Analyzer 发送的调查发现类型

IAM Access Analyzer 使用 Amazon 安全调查发现格式（ASFF）将调查发现发送到 Security Hub CSPM。在 ASFF 中，`Types` 字段提供调查发现类型。IAM Access Analyzer 的调查发现可能具有以下 `Types` 值。

- 外部访问调查发现 - 影响/数据暴露/授予的外部访问权限
- 外部访问调查发现 - 软件和配置检查/Amazon 安全最佳实践/授予的外部访问权限
- 未使用的访问调查发现 – 软件和配置检查/Amazon 安全最佳实践/未使用的权限
- 未使用的访问调查发现 - 软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 角色
- 未使用的访问调查发现 - 软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 用户密码
- 未使用的访问调查发现 - 软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 用户访问密钥

发送调查发现的延迟

IAM Access Analyzer 创建新的调查调查发现时，通常会在 30 分钟内将其发送到 Security Hub CSPM。但是，在极少数情况下，IAM Access Analyzer 可能不会收到有关策略更改的通知。例如：

- 对 Amazon S3 账户级别阻止公共访问设置的更改可能需要长达 12 小时才能反映在 IAM Access Analyzer 中。
- 在不更改基于资源的策略的情况下更改资源控制策略（RCP）不会触发对调查发现中报告的资源进行重新扫描。IAM Access Analyzer 会在下一个定期扫描期间（24 小时内）分析新策略或更新后的策略。
- 如果 Amazon CloudTrail 日志传输存在传输问题，则策略更改可能不会触发对调查发现中报告的资源进行重新扫描。

在这些情况下，IAM Access Analyzer 会在下一次定期扫描时分析新的或更新后的策略。

Security Hub CSPM 不可用时重试

如果 Security Hub CSPM 不可用，IAM Access Analyzer 会定期重试发送调查发现。

更新 Security Hub CSPM 中的现有调查发现

将调查发现发送到 Security Hub CSPM 后，IAM Access Analyzer 会继续将反映对调查发现活动的任何其他观测结果的更新发送到 Security Hub CSPM。这些更新反映在同一调查发现中。

对于外部访问调查发现，IAM Access Analyzer 按资源对其进行分组。在 Security Hub CSPM 中，如果 IAM Access Analyzer 中针对该资源的至少一项调查发现处于活动状态，则该资源的调查发现将保持活动状态。如果 IAM Access Analyzer 中某个资源的所有调查发现都已存档或解决，则 Security Hub CSPM 调查发现也会存档。当策略访问在公共访问和跨账户访问之间发生变化时，Security Hub CSPM 调查发现将更新。此更新可能包括对结果的类型、标题、描述和严重性的更改。

对于未使用访问调查发现，IAM Access Analyzer 不会按资源对其进行分组。相反，如果在 IAM Acccess Analyzer 中解决了未使用的访问调查发现，则 Security Hub CSPM 调查发现也将得到解决。当您更新生成未使用访问调查发现的 IAM 用户或角色时，Security Hub CSPM 调查发现也会更新。

在 Security Hub CSPM 中查看 IAM Access Analyzer 调查发现

要在 Security Hub CSPM 中查看 IAM Access Analyzer 调查发现，请在摘要页面的 Amazon: IAM Access Analyzer 部分中选择查看调查发现。或者，您可以从导航面板中选择 Findings（结果）。然后，您可以通过选择值为的 **`IAM Access Analyzer`** 的 Product name:（产品名称：）字段来筛选结果，以便仅显示 Amazon Identity and Access Management Access Analyzer 结果。

解释 Security Hub CSPM 中的 IAM Access Analyzer 调查发现名称

Amazon Identity and Access Management Access Analyzer 使用 Amazon 安全调查发现格式（ASFF）将调查发现发送到 Security Hub CSPM。在 ASFF 中，Types（类型）字段提供结果类型。ASFF 使用与 Amazon Identity and Access Management Access Analyzer 不同的命名方案。下表包含与 Security Hub CSPM 中显示的 Amazon Identity and Access Management Access Analyzer 调查发现关联的所有 ASFF 类型的详细信息。

ASFF 结果类型	Security Hub CSPM 调查发现标题	描述
效果/数据泄漏/外部访问已授权	<resource ARN> 允许公开访问	附加到资源的基于资源的策略允许所有外部主体对资源的公开访问。
软件和配置检查/Amazon 安全最佳实践/外部访问已授权	<resource ARN> 允许跨账户访问	附加到资源的基于资源的策略允许跨账户访问分析器信任区之外的外部主体。
软件和配置检查/Amazon 安全最佳实践/未使用的权限	<resource ARN> 包含未使用的权限	用户或角色包含未使用的服务和操作权限。

ASFF 结果类型	Security Hub CSPM 调查发现标题	描述
软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 角色	<resource ARN> 包含未使用的 IAM 角色	用户或角色包含未使用的 IAM 角色。
软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 用户密码	<resource ARN> 包含未使用的 IAM 用户密码	用户或角色包含未使用的 IAM 用户密码。
软件和配置检查/Amazon 安全最佳实践/未使用的 IAM 用户访问密钥	<resource ARN> 包含未使用的 IAM 用户访问密钥	用户或角色包含未使用的 IAM 用户访问密钥。

IAM Access Analyzer 的典型调查发现

IAM Access Analyzer 使用 Amazon 安全调查发现格式（ASFF）将调查发现发送到 Security Hub CSPM。

以下是 IAM Access Analyzer 针对外部访问调查发现的典型调查发现示例。

```
{
    "SchemaVersion": "2018-10-08",
    "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/my-analyzer/
arn:aws:s3:::amzn-s3-demo-bucket",
    "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
    "GeneratorId": "aws/access-analyzer",
    "AwsAccountId": "111122223333",
    "Types": ["Software and Configuration Checks/Amazon Security Best Practices/
External Access Granted"],
    "CreatedAt": "2020-11-10T16:17:47Z",
    "UpdatedAt": "2020-11-10T16:43:49Z",
    "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
    },
    "Title": "AwsS3Bucket/arn:aws:s3:::amzn-s3-demo-bucket/ allows cross-account
 access",
    "Description": "AWS::S3::Bucket/arn:aws:s3:::amzn-s3-demo-bucket/ allows cross-
account access from Amazon 444455556666",
```

```
    "Remediation": {
        "Recommendation": {"Text": "If the access isn't intended, it indicates a
 potential security risk. Use the console for the resource to modify or remove the
 policy that grants the unintended access. You can use the Rescan button on the Finding
 details page in the Access Analyzer console to confirm whether the change removed the
 access. If the access is removed, the status changes to Resolved."}
    },
    "SourceUrl": "https://console.aws.amazon.com/access-analyzer/home?region=us-
west-2#/findings/details/dad90d5d-63b4-6575-b0fa-ef9c556ge798",
    "Resources": [
        {
            "Type": "AwsS3Bucket",
            "Id": "arn:aws:s3:::amzn-s3-demo-bucket",
            "Details": {
                "Other": {
                    "External Principal Type": "AWS",
                    "Condition": "none",
                    "Action Granted": "s3:GetObject,s3:GetObjectVersion",
                    "External Principal": "444455556666"
                }
            }
        }
    ],
    "WorkflowState": "NEW",
    "Workflow": {"Status": "NEW"},
    "RecordState": "ACTIVE"
}
```

以下是 IAM Access Analyzer 针对未使用的访问调查发现的典型调查发现示例。

```
{
    "Findings": [
    {
      "SchemaVersion": "2018-10-08",
      "Id": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-
DO-NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "ProductArn": "arn:aws:securityhub:us-west-2::product/aws/access-analyzer",
      "ProductName": "IAM Access Analyzer",
      "CompanyName": "AWS",
      "Region": "us-west-2",
      "GeneratorId": "aws/access-analyzer",
      "AwsAccountId": "111122223333",
      "Types": [
```

```
        "Software and Configuration Checks/AWS Security Best Practices/Unused
 Permission"
      ],
      "CreatedAt": "2023-09-18T16:29:09.657Z",
      "UpdatedAt": "2023-09-21T20:39:16.651Z",
      "Severity": {
        "Product": 1,
        "Label": "LOW",
        "Normalized": 1
      },
      "Title": "AwsIamRole/arn:aws:iam::111122223333:role/IsengardRole-DO-NOT-DELETE/
 contains unused permissions",
      "Description": "AWS::IAM::Role/arn:aws:iam::111122223333:role/IsengardRole-DO-
NOT-DELETE/ contains unused service and action-level permissions",
      "Remediation": {
        "Recommendation": {
          "Text":"If the unused permissions aren’t required, delete the permissions to
 refine access to your account. Use the IAM console to modify or remove the policy that
 grants the unused permissions. If all the unused permissions are removed, the status
 of the finding changes to Resolved."
        }
      },
      "SourceUrl": "https://us-west-2.console.aws.amazon.com/access-analyzer/
home?region=us-west-2#/unused-access-findings?resource=arn%3Aaws%3Aiam%3A
%3A903798373645%3Arole%2FTestRole",
      "ProductFields": {
      "numberOfUnusedActions": "256",
      "numberOfUnusedServices": "15",
      "resourceOwnerAccount": "111122223333",
      "findingId": "DEMO24d8d-0d3f-4d3d-99f4-299fc8a62ee7",
      "findingType": "UnusedPermission",
      "aws/securityhub/FindingId": "arn:aws:securityhub:us-west-2::product/aws/access-
analyzer/arn:aws:access-analyzer:us-west-2:111122223333:analyzer/integTestAnalyzer-DO-
NOT-DELETE/arn:aws:iam::111122223333:role/TestRole/UnusedPermissions",
      "aws/securityhub/ProductName": "AM Access Analyzer",
      "aws/securityhub/CompanyName": "AWS"
    },
    "Resources": [
    {
      "Type": "AwsIamRole",
      "Id": "arn:aws:iam::111122223333:role/TestRole"
    }
  ],
  "WorkflowState": "NEW",
```

```
  "Workflow": {
    "Status": "NEW"
    },
  "RecordState": "ARCHIVED",
  "FindingProviderFields": {
    "Severity": {
      "Label": "LOW"
    },
    "Types": [
    "Software and Configuration Checks/AWS Security Best Practices/Unused Permission"
    ]
  }
  }
]
}
```

启用和配置集成

若要使用与 Security Hub CSPM 的集成，您必须启用 Security Hub CSPM。有关如何启用 Security Hub CSPM 的信息，请参阅《Amazon Security Hub User Guide》中的 Setting up Security Hub。

当您同时启用 IAM Access Analyzer 和 Security Hub CSPM 时，集成将自动启用。IAM Access Analyzer 立即开始将调查发现发送到 Security Hub CSPM。

如何停止发送调查发现

要停止向 Security Hub CSPM 发送调查发现，您可以使用 Security Hub CSPM 控制台或 API。

请参阅 Amazon Security Hub 用户指南中的禁用和启用来自集成的结果流（控制台）或禁用来自集成的结果流（Security Hub API、Amazon CLI）。

使用 Amazon CloudTrail 记录 IAM Access Analyzer API 调用

IAM Access Analyzer 已与 Amazon CloudTrail 集成，该服务提供 IAM Access Analyzer 中的用户、角色或 Amazon 服务采取的操作记录。CloudTrail 将 IAM Access Analyzer 的所有 API 调用作为事件捕获。捕获的调用包含来自 IAM Access Analyzer 控制台的调用和对 IAM Access Analyzer API 操作的代码调用。

如果您创建跟踪，则可以使 CloudTrail 事件持续传送到 Amazon S3 存储桶（包括 IAM Access Analyzer 的事件）。如果您不配置跟踪，则仍可在 CloudTrail 控制台中的事件历史记录中查看最新事件。

使用 CloudTrail 收集的信息，您可以确定向 IAM Access Analyzer 发出了什么请求、发出请求的 IP 地址、何人发出的请求、请求的发出时间以及其他详细信息。

要了解有关 CloudTrail 的更多信息，请参阅《Amazon CloudTrail 用户指南》。

CloudTrail 中的 IAM Access Analyzer 信息

在您创建 Amazon 账户时，将在该账户上启用 CloudTrail。当 IAM Access Analyzer 中发生活动时，该活动将记录在 CloudTrail 事件中，并与其他 Amazon 服务事件一同保存在 Event history（事件历史记录）中。您可以在 Amazon 账户中查看、搜索和下载最新事件。有关更多信息，请参阅使用 CloudTrail 事件历史记录查看事件。

要持续记录 Amazon 账户中的事件（包括 IAM Access Analyzer 的事件），请创建跟踪。通过跟踪记录，CloudTrail 可将日志文件传送至 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 Amazon 区域。此跟踪记录在 Amazon 分区中记录所有区域中的事件，并将日志文件传送至您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，进一步分析在 CloudTrail 日志中收集的事件数据并采取行动。有关更多信息，请参阅下列内容：

- 创建跟踪概述
- CloudTrail 支持的服务和集成
- 为 CloudTrail 配置 Amazon SNS 通知
- 从多个区域接收 CloudTrail 日志文件和从多个账户接收 CloudTrail 日志文件

CloudTrail 录入了所有 IAM Access Analyzer 操作，IAM Access Analyzer API 引用中记录了这些操作。例如，对 `CreateAnalyzer`、`CreateArchiveRule` 和 `ListFindings` 操作的调用会在 CloudTrail 日志文件中生成条目。

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 Amazon Identity and Access Management（IAM）用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

了解 IAM Access Analyzer 日志文件条目

跟踪记录是一种配置，可用于将事件作为日志文件传送到您指定的 Simple Storage Service（Amazon S3）存储桶。CloudTrail 日志文件包含一个或多个记录条目。一个事件表示来自任何源的一个请求，包括有关所请求的操作、操作的日期和时间、请求参数等方面的信息。CloudTrail 日志文件不是公有 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序显示。

下面的示例显示了一个 CloudTrail 日志条目，该条目说明了名为 `Alice-tempcreds` 的假定角色会话在“2021 年 6 月 14 日”执行的 `CreateAnalyzer` 操作。角色会话由名为 `admin-tempcreds` 的角色发出。

```
{
  "eventVersion": "1.05",
  "userIdentity":   {
    "type": "AssumedRole",
    "principalId": "AROAIBKEVSQ6C2EXAMPLE:Alice-tempcreds",
    "arn": "arn:aws:sts::111122223333:assumed-role/admin-tempcreds/Alice-tempcreds",
    "accountId": "111122223333",
    "accessKeyId": "AKIAIOSFODNN7EXAMPLE",
    "sessionContext": {
      "attributes": {
        "mfaAuthenticated": "true",
        "creationDate": "2021-06-14T22:54:20Z"
      },
      "sessionIssuer": {
        "type": "Role",
        "principalId": "AKIAI44QH8DHBEXAMPLE",
        "arn": "arn:aws:iam::111122223333:role/admin-tempcreds",
        "accountId": "111122223333",
        "userName": "admin-tempcreds"
      },
     "webIdFederationData": {},
    }
  },
  "eventTime": "2021-06-14T22:57:36Z",
  "eventSource": "access-analyzer.amazonaws.com",
  "eventName": "CreateAnalyzer",
  "awsRegion": "us-west-2",
  "sourceIPAddress": "198.51.100.179",
  "userAgent": "aws-sdk-java/1.12.79 Linux/5.4.141-78.230 OpenJDK_64-
Bit_Server_VM/25.302-b08 java/1.8.0_302 vendor/Oracle_Corporation cfg/retry-mode/
standard",
  "requestParameters": {
```

```
    "analyzerName": "test",
    "type": "ACCOUNT",
    "clientToken": "11111111-abcd-2222-abcd-222222222222",
        "tags": {
            "tagkey1": "tagvalue1"
        }
  },
  "responseElements": {
    "arn": "arn:aws:access-analyzer:us-west-2:111122223333:analyzer/test"
  },
  "requestID": "22222222-dcba-4444-dcba-333333333333",
  "eventID": "33333333-bcde-5555-bcde-444444444444",
  "readOnly": false,
  "eventType": "AwsApiCall",,
  "managementEvent": true,
  "recipientAccountId": "111122223333",
    "eventCategory": "Management"
}
```

IAM Access Analyzer 筛选条件键

您可以使用下面的筛选键定义存档规则（CreateArchiveRule）、更新存档规则（UpdateArchiveRule）、检索调查发现列表（ListFindings 和 ListFindingsV2），或检索资源（ListAccessPreviewFindings）的访问预览调查发现列表。使用 IAM API 或是 Amazon CloudFormation 来配置归档规则的操作没有区别。

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
资源	资源	ARN 唯一标识外部主体有权访问的资源。要了解更多信息，请参阅 Amazon 资源名称 (ARN)。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal 未使用

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
resourceType AWS::S3 Bucket | AWS::IAM :Role | AWS::SQS :Queue | AWS::Lambda::Function | AWS::Lambda::LayerVersion |AWS::KMS :Key | AWS::SecretsManager::Secret | AWS::EFS :FileSystem | AWS::EC2 :Snapshot |	资源类型	外部主体有权访问的资源类型。 **Note** 内部访问分析器并不支持外部访问分析器支持的所有资源类型。未使用的访问分析器仅支持 IAM 用户和角色。有关更多信息，请参阅 IAM Access Analyzer 支持的外部和内部访问资源类型。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal 未使用

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
AWS::ECR::Repository | AWS::RDS::DBSnapshot | AWS::RDS::DBClusterSnapshot | AWS::SNS::Topic | AWS::S3Express::DirectoryBucket | AWS::DynamoDB::Table | AWS::DynamoDB::Stream | AWS::IAM::User							

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
resourceOwnerAccount	资源所有者账户	拥有资源的 12 位数 Amazon 账户 ID。要了解更多信息，请参阅 Amazon 账户标识符。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal 未使用
isPublic	公有访问权限	指示结果是否报告具有允许公共访问的策略的资源。	布尔值	✓ 是	✓ 是	✓ 是	外部
findingType `ExternalAccess` | `UnusedIAMRole` | `UnusedIAMUserAccessKey` | `UnusedIAMUserPassword` | `UnusedPermission` | `InternalAccess`	调查发现类型	结果的类型。对于外部访问分析器，类型为 `ExternalAccess` 。对于未使用的访问分析器，类型可以是 `UnusedIAMRole` 、`UnusedIAMUserAccessKey` 、`UnusedIAMUserPassword` 或 `UnusedPermission` 。对于内部访问分析器，类型为 `InternalAccess` 。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal 未使用

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
resourceControlPolicyRestriction APPLIED | APPLICABLE | FAILED_EVALUATING_RCP | NOT_APPLICABLE	资源控制策略（RCP）限制	资源所有者通过 Organizations 资源控制策略（RCP）应用的限制类型。有关此筛选条件键值的更多信息，请参阅《IAM Access Analyzer API 参考》中的 ExternalAccessDetails 和 InternalAccessDetails。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal
serviceControlPolicyRestriction APPLIED | APPLICABLE | FAILED_EVALUATING_SCP | NOT_APPLICABLE	服务控制策略 (SCP) 限制	Organizations 服务控制策略 (SCP) 应用的限制类型。有关此筛选条件键值的更多信息，请参阅《IAM Access Analyzer API 参考》中的 InternalAccessDetails。	字符串	✓ 是	✓ 是	✓ 是	Internal

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
状态 `ACTIVE` | `ARCHIVED` | `RESOLVED`	状态	结果的当前状态。	字符串	⊗否	⊙ 是	⊙ 是	外部 Internal 未使用
error	错误	指示为结果报告的错误。	字符串	⊙ 是	⊙ 是	⊙ 是	外部 Internal
principal.AWS	Amazon 账户	在结果的 `Principal` 字段中授予对资源访问权限的账户。输入外部 Amazon 用户或角色的 12 位数 Amazon 账户 ID 或 ARN。要了解更多信息，请参阅 Amazon 账户标识符。	字符串	⊙ 是	⊙ 是	⊙ 是	外部
principal.Federated	联合用户	有权访问结果中资源的联合身份的 ARN。要了解更多信息，请参阅身份提供商和联合身份验证。	字符串	⊙ 是	⊙ 是	⊙ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
condition .aws:PrincipalArn	主体 ARN	指定作为资源访问条件的主体（IAM 用户、角色或组）的 ARN。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition .aws:PrincipalOrgID	主体组织 ID	指定作为资源访问条件的主体的组织标识符。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition .aws:PrincipalOrgPaths	主体组织路径	指定作为资源访问条件的组织或组织单元 (OU) 的 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition .aws:SourceIp	源 IP	允许主体在使用指定 IP 地址时访问资源的 IP 地址。要了解更多信息，请参阅 Amazon 全局条件上下文键。	IP 地址	✓ 是	✓ 是	✓ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
condition .aws:SourceVpc	源 VPC	允许主体在使用指定 VPC 时访问资源的 VPC ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition .aws:UserId	用户 ID	外部账户中 IAM 用户的用户 ID，该用户指定作为访问资源的条件。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition .aws:VpceAccount	VPCE 账户	允许主体访问资源的 VPC 端点的账户 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal
condition .aws:SourceVpcArn	源 VPC ARN	允许主体在使用指定 VPC 时访问资源的 VPC ARN。要了解更多信息，请参阅 Amazon 全局条件上下文键。	ARN	✓ 是	✓ 是	✓ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
condition.aws:vpceorgid	VPCE 组织 ID	允许主体访问资源的 VPC 端点的组织 ID。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部 Internal
condition.aws:VpceOrgPaths	VPCE 组织路径	允许主体访问资源的 VPC 端点的组织单元 (OU)。要了解更多信息，请参阅 Amazon 全局条件上下文键。	字符串（列表）	✓ 是	✓ 是	✓ 是	外部 Internal
condition.cognito-identity.amazonav.com:aud	Cognito 受众	Amazon Cognito 身份池 ID，指定作为结果中 IAM 角色访问的条件。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
condition.graph.facebook.com:app_id	Facebook 应用程序 ID	将 Facebook 应用程序 ID（或网站 ID）指定作为条件，用于允许对结果中 IAM 角色的“用 Facebook 登录”联合身份访问。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition.accounts.google.com:aud	Google 受众	指定作为访问 IAM 角色的条件的 Google 应用程序 ID。要了解更多信息，请参阅 IAM 和 Amazon STS 条件上下文键。	字符串	✓ 是	✓ 是	✓ 是	外部
condition.kms:CallerAccount	KMS 密钥 ID	拥有调用实体的 Amazon 账户 ID（IAM 用户、角色或根用户），调用 Amazon KMS 的服务使用该账户。要了解更多信息，请参阅 Amazon Key Management Service 的条件键。	字符串	✓ 是	✓ 是	✓ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
condition.www.amazon.com:app_id	Amazon 应用程序 ID	将 Amazon 应用程序 ID（或网站 ID）指定作为条件，用于允许对角色的“Login with Amazon”联合身份访问。要了解更多信息，请参阅	字符串	✓ 是	✓ 是	✓ 是	外部
id	调查发现 ID	结果的 ID。	字符串	✗否	✓ 是	✓ 是	外部 Internal 未使用
changeType CHANGED | NEW | UNCHANGE		提供有关访问预览查找结果如何与 IAM Access Analyzer 中标识的现有访问进行比较的上下文。	字符串	✗否	✗否	✓ 是	外部
existingFindingId		IAM Access Analyzer 中查找结果的现有 ID，仅为访问预览中的现有查找结果提供。	字符串	✗否	✗否	✓ 是	外部

Criterion	Amazon Web Services 管理控制台字段	描述	Type	存档规则	列出结果	列出访问预览结果	支持的分析器类型
existingFindingStatus		查找结果的现有状态，仅为访问预览中的现有调查结果提供。	字符串	⊗否	⊗否	✓ 是	外部

将服务关联角色用于 Amazon Identity and Access Management Access Analyzer

Amazon Identity and Access Management Access Analyzer 使用 IAM 服务相关角色。服务相关角色是一种独特的 IAM 角色类型，直接关联到 IAM Access Analyzer。服务相关角色由 IAM Access Analyzer 预定义，并包含该功能代表您调用其他 Amazon 服务所需的一切权限。

服务相关角色让您可以轻松设置 IAM Access Analyzer，因为您不必手动添加必要的权限。IAM Access Analyzer 定义其服务相关角色的权限，除非另外定义，否则只有 IAM Access Analyzer 可以代入其角色。定义的权限包括信任策略和权限策略，而且权限策略不能附加到任何其他 IAM 实体。

有关支持服务相关角色的其他服务的信息，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色列中显示为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

的服务关联角色权限Amazon Identity and Access Management Access Analyzer

Amazon Identity and Access Management Access Analyzer 使用名为 AWSServiceRoleForAccessAnalyzer 的服务相关角色：允许 Access Analyzer 分析外部访问的资源元数据，并分析活动以识别未使用的访问。

AWSServiceRoleForAccessAnalyzer 服务相关角色信任以下服务来代入该角色：

- `access-analyzer.amazonaws.com`

名为 `AccessAnalyzerServiceRolePolicy` 的角色权限策略允许 IAM Access Analyzer 完成对特定资源的操作。

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务关联角色。有关更多信息，请参阅《IAM 用户指南》中的服务关联角色权限。

创建 IAM Access Analyzer 的服务相关角色

您无需手动创建服务关联角色。当您在 Amazon Web Services 管理控制台 或 Amazon API 中启用 IAM Access Analyzer 时，IAM Access Analyzer 将为您创建服务相关角色。在您启用 IAM Access Analyzer 的所有区域中使用相同的服务相关角色。外部访问和未使用的访问调查发现使用相同的服务相关角色。

> **Note**
>
> IAM Access Analyzer 是区域性的。您必须在每个区域内单独启用 IAM Access Analyzer。

如果您删除此服务相关角色，则在您下次创建分析器时，IAM Access Analyzer 会重新创建该角色。

您也可以使用 IAM 控制台为 Access Analyzer 使用案例创建服务相关角色。在 Amazon CLI 或 Amazon API 中，使用 `access-analyzer.amazonaws.com` 服务名称创建服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的创建服务相关角色。如果您删除了此服务相关角色，可以使用同样的过程再次创建角色。

编辑 IAM Access Analyzer 的服务相关角色

IAM Access Analyzer 不允许编辑 AWSServiceRoleForAccessAnalyzer 服务相关角色。创建服务关联角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。但是可以使用 IAM 编辑角色描述。有关更多信息，请参阅《IAM 用户指南》中的编辑服务关联角色。

删除 IAM Access Analyzer 的服务相关角色

如果您不再需要使用某个需要服务相关角色的特征或服务，我们建议您删除该角色。这样您就没有未被主动监控或维护的未使用实体。但是，您必须先清除服务相关角色的资源，然后才能手动删除它。

如果 Amazon Organizations 中的一个或多个区域启用了 IAM Access Analyzer，则必须先删除组织所有区域中的所有分析器，然后才能尝试删除此角色。

> **Note**
>
> 尝试删除资源时，如果 IAM Access Analyzer 正在使用该角色，删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

要删除 AWSServiceRoleForAccessAnalyzer 角色使用的 IAM Access Analyzer 资源

1. 通过 https://console.aws.amazon.com/iam/ 打开 IAM 控制台。
2. 在 Access reports (访问报告) 部分中的 Access analyzer (访问分析器) 下，选择 Analyzers (分析器)。
3. 选择要从中删除附加到服务相关角色的 IAM Access Analyzer 资源的分析器。
4. 选择删除。
5. 要确认您要删除分析器，请输入 **`delete`**，然后选择 Delete (删除)。

使用 IAM 手动删除服务关联角色

使用 IAM 控制台、Amazon CLI 或Amazon API 删除 AWSServiceRoleForAccessAnalyzer 服务相关角色。有关更多信息，请参见《IAM 用户指南》中的删除服务相关角色。

IAM Access Analyzer 服务相关角色的支持区域

IAM Access Analyzer 支持在服务可用的所有区域使用服务相关角色。有关更多信息，请参阅 Amazon 区域和端点。

预览访问

除有助于您识别与外部实体共享的资源外，Amazon IAM Access Analyzer 还会在部署资源权限之前显示 IAM Access Analyzer 调查发现预览，以便您可以验证策略更改是否仅授予对资源的预期公共和跨账户存取权限。这可以帮助您从预期的外部访问资源开始操作。

您可以预览和验证对 Amazon S3 控制台中 Amazon S3 存储桶的公共和跨账户访问权限。您还可以通过为您的资源提供建议的权限，以使用 IAM Access Analyzer API 预览 Amazon S3 存储桶、Amazon KMS 密钥、IAM 角色、Amazon SQS 队列和 Secrets Manager 密钥的公共和跨账户访问权限。

主题

- 在 Amazon S3 控制台中预览访问权限
- 使用 IAM Access Analyzer API 预览访问

在 Amazon S3 控制台中预览访问权限

在 Amazon S3 控制台中完成存储桶策略后，您可以选择预览 Amazon S3 存储桶的公共和跨账户访问权限。您可以验证策略更改是否仅授予预期的外部访问权限，然后再选择 Save changes（保存

更改）。利用此可选步骤，您可以预览存储桶的 Amazon Identity and Access Management Access Analyzer 结果。您可以验证策略更改是否引入了新的结果，或是解析现有的结果以供外部访问。您可以跳过此验证步骤，并随时保存您的 Amazon S3 存储桶策略。

要预览对存储桶的外部访问权限，您必须在存储桶的区域中拥有一个活动账户分析器，并且该账户需作为信任区域。您还必须具有使用 IAM Access Analyzer 和预览访问所需的权限。有关启用 IAM Access Analyzer 以及所需权限的更多信息，请参阅 Amazon Identity and Access Management Access Analyzer 入门。

要在创建或编辑存储桶策略时预览 Amazon S3 存储桶的访问

1. 完成创建或编辑存储桶策略后，请确保您的策略是有效的 Amazon S3 存储桶策略。策略 ARN 必须与存储桶 ARN 相匹配，且策略元素必须有效。
2. 在政策下方的 Preview external access（预览外部访问）项下，选择活动的账户分析器，然后选择 Preview（预览）。系统将为您的存储桶生成 IAM Access Analyzer 结果的预览。预览会分析显示的 Amazon S3 存储桶策略以及现有存储桶权限。这包括存储桶和账户 BPA 设置、存储桶 ACL、Amazon S3 访问点和附加到存储桶的多区域访问点，以及它们的策略和 BPA 设置。
3. 访问预览完成后，将显示 IAM Access Analyzer 结果的预览。在您保存策略后，每个结果都会报告账户外的主体实例，该实例可访问您的存储桶。您可以通过查看每个结果来验证对存储桶的访问权限。结果标题提供访问的摘要，您可以展开结果以查看结果详细信息。结果标记可提供有关保存存储桶策略如何更改对存储桶的访问权限的上下文。例如，它们可帮助您验证策略更改是否引入了新的结果，或是解决了外部访问的现有结果：

   a. 新结果 – 表示策略将引入的新外部访问结果。

   b. 已解决 – 表示策略将删除的现有外部访问结果。

   c. 已归档 — 表示将根据分析器的归档规则自动归档的新外部访问结果，这些规则定义了何时应将结果标记为预期结果。

   d. 现有 – 表示将保持不变的外部访问的现有结果。

   e. 公开 — 如果一个结果是用于公共访问资源，它将带有一个 Public（公开）徽章，此外还将带有上述徽章之一。
4. 如果您识别不打算引入或删除的外部访问，则可以修改策略，然后选择 Preview（预览），直到已经达到您想要的外部访问。如果你有一个标记标注为 Public（公开）的结果，我们建议您修改策略以删除公共访问权限，然后再选择 Save changes（保存更改）。预览访问是一个可选步骤，您随时都可以选择 Save changes（保存更改）。

使用 IAM Access Analyzer API 预览访问

您可以使用 IAM Access Analyzer API 预览 Amazon S3 存储桶的公共和跨账户访问、Amazon KMS 密钥、IAM 角色、Amazon SQS 队列和 Secrets Manager 密钥。您可以通过为您拥有的现有资源或您想要部署的新资源提供建议的权限来预览访问。

要预览对资源的外部访问，您必须具有适用于资源账户和区域的活动账户分析器。您还必须具有使用 IAM Access Analyzer 和预览访问所需的权限。有关启用 IAM Access Analyzer 以及所需权限的更多信息，请参阅 Amazon Identity and Access Management Access Analyzer 入门。

要预览资源的访问，可以使用 `CreateAccessPreview` 操作，并提供分析器 ARN 和资源的访问控制配置。该服务返回访问预览的唯一 ID，您可以使用该 ID 通过 `GetAccessPreview` 操作查看访问预览的状态。当状态为 `Completed`，则可以使用 `ListAccessPreviewFindings` 操作来检索为访问预览生成的结果。`GetAccessPreview` 和 `ListAccessPreviewFindings` 操作将检索大约 24 小时内创建的访问预览和结果。

检索的每个结果都包含描述访问的结果详细信息。结果的预览状态，描述权限部署之后结果是否为 `Active`、`Archived` 或者 `Resolved`，以及 `changeType`。`changeType` 提供了有关访问预览结果如何与 IAM Access Analyzer 中标识的现有访问进行比较的上下文：

- 新结果 – 结果适用于新引入的访问。
- 不变 – 预览结果是将保持不变的现有结果。
- 已更改 – 预览结果是状态发生变化的现有查找结果。

`status` 和 `changeType` 帮助您了解资源配置将如何更改现有资源访问。如果 `changeType` 为 `Unchanged` 或“已更改”，则结果还将包含 IAM Access Analyzer 中结果的现有 ID 和状态。例如，`Changed` 结果具有预览状态 `Resolved` 和现有状态 `Active`，指示资源的现有 `Active` 结果将由于建议权限更改而成为 `Resolved`。

您可以使用 `ListAccessPreviews` 操作来检索指定分析器的访问预览列表。此操作将检索约一小时内创建的访问预览信息。

通常，如果访问预览适用于现有资源，并且未指定配置选项，则预设情况下，访问预览将使用现有资源配置。如果访问预览适用于新资源，并且未指定配置选项，则访问预览将使用默认值，具体取决于资源类型。有关每个资源类型的配置案例，请参阅下文。

预览对 Amazon S3 存储桶的访问

要为新 Amazon S3 存储桶或您拥有的现有 Amazon S3 存储桶创建访问预览，您可以通过指定附加到存储桶的 Amazon S3 存储桶策略、存储桶 ACL、存储桶 BPA 设置和 Amazon S3 访问点（包括多区域访问点）来建议存储桶配置。

> **Note**
>
> 在尝试为新存储桶创建访问预览之前，我们建议您调用 Amazon S3 HeadBucket 操作来检查指定的存储桶是否已经存在。此操作可用于确定存储桶是否存在以及您是否有权访问该存储桶。

存储桶策略 — 如果配置适用于现有 Amazon S3 存储桶，并且您未指定 Amazon S3 存储桶策略，则访问预览将使用附加到存储桶的现有策略。如果访问预览适用于新资源，并且您没有指定 Amazon S3 存储桶策略，则访问预览会假定存储桶没有策略。要建议删除现有存储桶策略，您可以指定一个空字符串。有关支持的存储桶策略限制的更多信息，请参阅存储桶策略示例。

存储桶 ACL 授权 — 每个存储桶最多可提议 100 个 ACL 授权。如果建议的授权配置适用于现有存储桶，则访问预览会使用建议的授权配置列表来代替现有授权。否则，访问预览将使用存储桶的现有授权。

存储桶访问点 — 分析支持每个存储桶多达 100 个访问点，包括多区域访问点，包括每个存储桶最多可建议的 10 个新访问点。如果建议的 Amazon S3 访问点配置适用于现有存储桶，则访问预览会使用建议的访问点配置代替现有访问点。要建议没有策略的访问点，可以提供空字符串作为访问点策略。有关访问点策略限制的更多信息，请参阅访问点限制和局限性。

阻止公有访问配置 — 如果建议的配置适用于现有 Amazon S3 存储桶，并且您未指定配置，则访问预览将使用现有设置。如果建议的配置适用于新存储桶，并且您未指定存储桶 BPA 配置，则访问预览会使用 `false`。如果建议的配置适用于新的访问点或多区域访问点，并且您未指定访问点 BPA 配置，则访问预览会使用 `true`。

预览对您的 Amazon KMS 密钥的访问

要为新的 Amazon KMS 密钥或您所拥有的现有 Amazon KMS 密钥创建访问预览，可以通过指定免邮策略和 Amazon KMS 授权配置来建议 Amazon KMS 密钥。

Amazon KMS 密钥策略 — 如果配置适用于现有密钥，并且您未指定密钥策略，则访问预览将使用该密钥的现有策略。如果访问预览适用于新资源，并且您未指定密钥策略，则访问预览将使用默认密钥策略。建议的密密钥策略不能是空字符串。

Amazon KMS 授权 — 分析最多支持每个配置 100 个 KMS 授权 *。* 如果建议的授权配置适用于现有密钥，则访问预览会在现有授权中使用建议的授权配置列表。否则，访问预览将使用该密钥的现有授权。

预览对 IAM 角色的访问

要为新 IAM 角色或您拥有的现有 IAM 角色创建访问预览，您可以通过指定信任策略来建议 IAM 角色配置。

角色信任策略 — 如果配置适用于新 IAM 角色，则必须指定信任策略。如果配置适用于您拥有的现有 IAM 角色，并且您没有建议信任策略，则访问预览将使用角色的现有信任策略。建议的信任策略不能是空字符串。

预览对您的 Amazon SQS 队列的访问

要为新 Amazon SQS 队列或您拥有的现有 Amazon SQS 队列创建访问预览，您可以通过为队列指定 Amazon SQS 策略来建议 Amazon SQS 队列配置。

Amazon SQS 队列策略 — 如果配置适用于现有 Amazon SQS 队列，并且您未指定 Amazon SQS 策略，则访问预览将为队列使用现有的 Amazon SQS 策略。如果访问预览适用于新资源，并且您没有指定策略，则访问预览会假定 Amazon SQS 队列没有策略。要建议删除现有 Amazon SQS 队列策略，您可以为 Amazon SQS 策略指定一个空字符串。

预览对您的 Secrets Manager 密钥的访问

要为新的 Secrets Manager 密钥或您拥有的现有 Secrets Manager 密钥创建访问预览，您可以通过指定密钥策略和可选的 Amazon KMS 加密密钥。

密钥策略 — 如果配置适用于现有密钥，并且您未指定密钥策略，则访问预览将使用现有的密钥策略。如果访问预览适用于新资源，并且您未指定策略，则访问预览会假定密钥没有策略。要建议删除现有策略，可以指定一个空字符串。

Amazon KMS 加密密钥 — 如果建议的配置适用于新密钥，并且您没有指定 Amazon KMS 密钥 ID，则访问预览将使用 Amazon 账户的默认 KMS 密钥。如果您为 Amazon KMS 密钥 ID 指定空字符串，则访问预览将使用 Amazon 账户的默认 KMS 密钥。

检查验证策略

IAM Access Analyzer 提供策略检查，帮助您在将 IAM policy 附加到实体之前对其进行验证。其中包括由策略验证提供的基本策略检查，以根据策略语法和 Amazon 最佳实践来验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。

您可以使用自定义策略检查，根据安全标准检查新的访问。每次检查新的访问都会产生费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

自定义策略检查的工作原理

您可以使用 Amazon Identity and Access Management Access Analyzer 自定义策略检查，根据指定的安全标准验证策略。您可以运行下列类型的自定义策略检查：

- 对照参考策略检查：编辑策略时，您可以检查与参考策略（如现有版本的策略）相比，更新后的策略是否授予新的访问。在 IAM 控制台中使用 Amazon Command Line Interface（Amazon CLI）、IAM Access Analyzer API（API）或 JSON 策略编辑器编辑策略时，可以运行此检查。

  > **Note**
  >
  > IAM Access Analyzer 自定义策略检查允许在 `Principal` 元素中使用通配符作为参考资源策略。

- 对照 IAM 操作或资源列表进行检查：您可以检查以确保策略不允许特定 IAM 操作或资源。如果只指定了操作，则 IAM Access Analyzer 会检查是否对策略中的所有资源具有操作访问权限。如果只指定了资源，则 IAM Access Analyzer 会检查哪些操作可以访问指定的资源。如果同时指定了操作和资源，则 IAM Access Analyzer 会检查哪些指定操作可以访问指定的资源。使用 Amazon CLI 或 API 创建或编辑策略时，可以运行此检查。
- 检查是否具有公共访问权限：您可以检查资源策略是否可以授予对指定的资源类型的公共访问权限。使用 Amazon CLI 或 API 创建或编辑策略时，可以运行此检查。这种类型的自定义策略检查与预览访问权限不同，因为该检查不需要任何账户或外部访问分析器上下文。访问预览允许您在部署资源权限之前预览 IAM Access Analyzer 调查发现，而自定义检查确定策略是否可以授予公共访问权限。

每次自定义策略检查都会产生费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

您可以对基于身份和基于资源的策略运行自定义策略检查。自定义策略检查不依赖模式匹配技术或检查访问日志来确定策略是否允许新的访问或指定访问。与外部访问调查发现类似，自定义策略检查是基于 Zelkova 构建的。Zelkova 将 IAM policy 转换为等效逻辑语句，并针对问题运行一套通用和专门逻辑求解器（可满足性模理论）。为了检查新的访问或指定访问，IAM Access Analyzer 将 Zelkova 重复应用于策略。查询变得越来越具体，可以根据策略的内容来表征策略允许的行为类别。有关可满足性模理论的更多信息，请参阅可满足性模理论。

在极少数情况下，IAM Access Analyzer 无法完全确定策略语句是授予新的访问权限还是指定访问权限。在这种情况下，如果自定义策略检查失败，就会导致声明误报。IAM Access Analyzer 旨在提供全

面的策略评估，并尽量减少误报。这种方法意味着 IAM Access Analyzer 提供了高度的保证，即通过检查意味着策略未授予访问权限。您可以查看 IAM Access Analyzer 响应中报告的策略语句，手动检查失败的检查。

检查新访问的参考策略的示例

您可以在 GitHub 上的 IAM Access Analyzer 自定义策略检查示例存储库中找到参考策略示例，了解如何为新的访问设置和运行自定义策略检查。

> **在使用这些示例之前**
>
> 在使用这些示例参考策略之前，请执行以下操作：
>
> - 仔细查看并根据您的独特需求自定义参考策略。
> - 通过您所用的 Amazon Web Services 服务 在您的环境中全面测试参考策略。
>
> 参考策略演示了自定义策略检查的实现和使用。这些示例策略并不是要完全按照所示实施的官方Amazon建议或最佳实践。您有责任仔细测试参考策略是否适合满足您环境中的安全要求。
> - 自定义策略检查的分析与环境无关。其分析只考虑输入策略中包含的信息。例如，自定义策略检查无法检查账户是否是特定 Amazon 组织的成员。因此，自定义策略检查无法根据 `aws:PrincipalOrgId` 和 `aws:PrincipalAccount` 条件键的条件键值来比较新的访问。

检查失败的自定义策略检查

当自定义策略检查失败时，IAM Access Analyzer 的响应包括导致检查失败的策略语句的语句 ID（`Sid`）。虽然语句 ID 是一个可选的策略元素，但我们建议您为每个策略语句添加一个语句 ID。自定义策略检查还会返回语句索引，以帮助确定检查失败的原因。语句索引遵循从零开始的编号，其中第一条语句的编号为 0。当有多个语句导致检查失败时，检查一次只返回一个语句 ID。我们建议您修复原因中突出显示的语句，然后重新运行检查，直到检查通过。

使用 IAM Access Analyzer 验证策略

您可以使用 Amazon Identity and Access Management Access Analyzer 策略验证来验证您的策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。IAM Access Analyzer 将根据 IAM policy 语法和 Amazon 最佳实践来验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些结果提供了可操作的建议，

可帮助您编写可操作且符合安全最佳实践的策略。要查看 IAM Access Analyzer 运行的基本策略检查列表，请参阅 IAM 策略验证检查引用。

在 IAM（控制台）中验证策略

在 IAM 控制台中创建或编辑托管策略时，可以查看 IAM Access Analyzer 策略验证生成的调查发现。您还可以查看这些内联用户或角色策略的结果。IAM Access Analyzer 不会为内联组策略生成这些调查发现。

查看 IAM JSON 策略的策略检查生成的结果

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 使用以下方法之一开始创建或编辑策略：
   a. 要创建新的托管策略，请转到 Policies（策略）页面并创建新策略。有关更多信息，请参阅 使用 JSON 编辑器创建策略。
   b. 要查看现有客户管理型策略的策略检查，请转到策略页面，选择策略的名称，然后选择编辑。有关更多信息，请参阅 编辑客户托管策略（控制台）。
   c. 要查看针对用户或角色的内联策略的策略检查，请转到用户或角色页面，选择用户或角色的名称，在权限选项卡上选择策略名称，然后选择编辑。有关更多信息，请参阅 编辑内联策略（控制台）。
3. 在策略编辑器中，选择 JSON 选项卡。
4. 在策略下方的策略验证窗格中，选择以下一个或多个选项卡。选项卡名称还指示策略的每种查找类型的数量。
   - Security（安全）— 如果您的策略允许访问，但由于访问权限过于宽松而致使 Amazon 认为存在安全风险，则可查看相关警告。
   - Errors（错误）— 如果策略包含阻止策略运行的行，则可查看错误。
   - 警告 – 如果您的策略不符合最佳实践，但问题不属于安全风险，则可查看警告。
   - Suggestions（建议）— 如果 Amazon 的建议不影响策略权限的改进，则可查看建议。
5. 查看 IAM Access Analyzer 策略检查提供的结果详细信息。每个结果都会指示所报告问题的位置。要了解有关导致问题的原因以及如何解决问题的详细信息，请选择结果旁的 Learn more（了解更多）链接。您还可以在 Access Analyzer policy checks（Access Analyzer 策略检查）参考页面搜索与各个结果关联的策略检查。
6. 可选。如果正在编辑现有策略，则可以运行自定义策略检查，以确定与现有版本相比，更新后的策略是否授予新的访问权限。在策略下方的策略验证窗格中，选择检查新访问选项卡，然后选择检查

策略。如果修改后的权限授予新的访问权限，则该语句将在策略验证窗格中突出显示。如果不打算授予新的访问权限，请更新策略声明并选择检查策略，直到检测不到新的访问。有关更多信息，请参阅 使用 IAM Access Analyzer 自定义策略检查来验证策略。

> **Note**
>
> 每次检查新的访问都会产生费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

7. 请更新策略以解决结果。

> **Important**
>
> 在生产工作流中实施新策略之前，测试新策略或彻底编辑这些策略。

8. 完成后，选择下一步。策略验证器会报告 IAM Access Analyzer 未报告的任何语法错误。

> **Note**
>
> 您可以随时在可视化和 JSON 选项卡之间切换。但如果您进行更改或在可视化选项卡中选择下一步，IAM 可能会重组您的策略，以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

9. 对于新策略，在查看和创建页面上，输入您要创建的策略的策略名称和描述（可选）。查看此策略中定义的权限，了解策略授予的权限。然后，选择创建策略以保存您的工作。

   对于现有策略，在查看和保存页面上，查看此策略中定义的权限，了解策略授予的权限。选择将此新版本设置为默认版本。复选框以将更新后的版本保存为策略的默认版本。然后选择保存更改以保存您的工作。

使用 IAM Access Analyzer（Amazon CLI 或 Amazon API）验证策略

您可以从 Amazon Command Line Interface（Amazon CLI）查看 IAM Access Analyzer 策略验证生成的调查发现。

要查看 IAM Access Analyzer 策略验证（Amazon CLI 或 Amazon API）生成的调查发现

使用以下值之一：

- Amazon CLI: aws accessanalyzer validate-policy
- Amazon API: ValidatePolicy

IAM 策略验证检查引用

您可以使用 Amazon Identity and Access Management Access Analyzer 策略验证来验证您的策略。您可以在 IAM 控制台中使用 Amazon CLI、Amazon API 或 JSON 策略编辑器创建或编辑策略。IAM Access Analyzer 将根据 IAM policy 语法和 Amazon 最佳实践来验证您的策略。您可以查看策略验证检查结果，其中包括策略的安全警告、错误、常规警告和策略建议。这些结果提供了可操作的建议，可帮助您编写可操作且符合安全最佳实践的策略。IAM Access Analyzer 提供的基本策略检查列表如下。运行策略验证检查不收取额外费用。要了解有关使用策略验证来验证策略的更多信息，请参阅 使用 IAM Access Analyzer 验证策略。

错误 — 不允许 ARN 账户

问题代码：ARN_ACCOUNT_NOT_ALLEVED

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
ARN account not allowed: The service {{service}} does not support specifying an account
 ID in the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service {{service}} does not support specifying an account ID in
 the resource ARN."
```

解决错误

请从资源 ARN 中删除账户 ID。某些 Amazon 服务的资源 ARN不支持指定账户 ID。

例如，Amazon S3 不支持将账户 ID 作为存储桶 ARN 中的命名空间。Amazon S3 存储桶名称是全局唯一的，并且命名空间由所有 Amazon 账户共享。要查看 Amazon S3 中所有可用的资源类型，请参阅服务授权参考中的 Amazon S3 定义的资源类型。

相关术语

- 策略资源
- 账户标识符
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 不允许 ARN 区域

问题代码：ARN_REGION_NOT_ALLOWED

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
ARN Region not allowed: The service {{service}} does not support specifying a Region in
 the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service {{service}} does not support specifying a Region in the
 resource ARN."
```

解决错误

请从资源 ARN 中删除区域。某些 Amazon 服务的资源 ARN不支持指定区域。

例如，IAM 是一项全球性服务。IAM 资源 ARN 的区域部分始终保留为空。IAM 资源是全球性的，如同现在的 Amazon 账户。例如，以 IAM 用户身份登录后，您可以在任何地理区域访问 Amazon 服务。

- 策略资源
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 数据类型不匹配

问题代码：DATA_TYPE_MISMATCH

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Data type mismatch: The text does not match the expected JSON data type {{data_type}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The text does not match the expected JSON data type {{data_type}}."
```

解决错误

请更新文本以使用支持的数据类型。

例如，`Version` 全局条件键需要 `String` 数据类型。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- 全局条件键
- IAM JSON 策略元素：条件运算符

错误 — 使用不同的大小写重复密钥

问题代码：DUPLICATE_KEYS_WITH_DIFFERENT_CASE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Duplicate keys with different case: The condition key {{key}} appears more than once
 with different capitalization in the same condition block. Remove the duplicate
 condition keys.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{key}} appears more than once with different
 capitalization in the same condition block. Remove the duplicate condition keys."
```

解决错误

查看同一条件数据块中的类似条件键，并对所有实例使用相同的大写字母。

条件数据块是策略语句 Condition 元素中的文本。条件键名称 不区分大小写。条件键值是否区分大小写取决于您使用的条件运算符。有关条件键中区分大小写的更多信息，请参阅 IAM JSON 策略元素：Condition。

相关术语

- 条件
- 条件数据块
- 全局条件键
- Amazon 服务条件键

错误 — 无效的操作

问题代码：INVALID_ACTION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid action: The action {{action}} does not exist. Did you mean {{valid_action}}?
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action {{action}} does not exist. Did you mean
 {{valid_action}}?"
```

解决错误

指定的操作无效。如果您键入了错误的服务前缀或操作名称，则可能会发生这种情况。对于一些常见问题，策略检查将返回建议的操作。

相关术语

- 策略操作
- Amazon 服务操作

出现此错误的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其策略语句中包含无效操作。无效操作不会影响策略所授予的权限。使用 Amazon 托管策略作为创建托管策略的参考时，Amazon 会建议您从策略中删除无效操作。

- AmazonEMRFullAccessPolicy_v2
- CloudWatchSyntheticsFullAccess

错误 — 无效的 ARN 账户

问题代码：INVALID_ARN_ACCOUNT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid ARN account: The resource ARN account ID {{account}} is not valid. Provide a
 12-digit account ID.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN account ID {{account}} is not valid. Provide a 12-
digit account ID."
```

解决错误

请更新资源 ARN 中的账户 ID。账户 ID 是 12 位整数。要了解如何查看您的账户 ID，请参阅查找您的 Amazon 账户 ID。

相关术语

- 策略资源
- 账户标识符
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 无效的 ARN 前缀

问题代码：INVALID_ARN_PREFIX

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid ARN prefix: Add the required prefix (arn) to the resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add the required prefix (arn) to the resource ARN."
```

解决错误

Amazon 资源 ARN 必须包含所需 `arn:` 前缀。

相关术语

- 策略资源
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 无效的 ARN 区域

问题代码：INVALID_ARN_REGION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid ARN Region: The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Region {{region}} is not valid for this resource. Update the
 resource ARN to include a supported Region."
```

解决错误

资源类型在指定区域不支持。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅区域列表。

相关术语

- 策略资源
- 资源 ARN
- 区域名称和代码

错误 — 无效的 ARN 资源

问题代码：INVALID_ARN_RESOURCE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid ARN resource: Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Resource ARN does not match the expected ARN format. Update the
 resource portion of the ARN."
```

解决错误

资源 ARN 必须与已知资源类型的规范相匹配。要查看服务的预期 ARN 格式，请参阅Amazon 服务的操作、资源和条件键。请选择服务的名称以查看其资源类型和 ARN 格式。

相关术语

- 策略资源

- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 无效的 ARN 服务案例

问题代码：INVALID_ARN_SERVICE_CASE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid ARN service case: Update the service name {{service}} in the resource ARN to
 use all lowercase letters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Update the service name {{service}} in the resource ARN to use all
 lowercase letters."
```

解决错误

资源 ARN 中的服务必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅Amazon 服务的操作、资源和条件键。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- 策略资源
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 无效的条件数据类型

问题代码：INVALID_CONDION_DATA_TYPE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid condition data type: The condition value data types do not match. Use condition
 values of the same JSON data type.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition value data types do not match. Use condition values of
 the same JSON data type."
```

解决错误

条件键值对中的值必须与条件键和条件运算符的数据类型匹配。要查看服务的条件键数据类型，请参阅Amazon 服务的操作、资源和条件键。请选择服务名称以查看该服务的条件键。

例如，`CurrentTime` 全局条件键支持 `Date` 条件运算符。如果为条件数据块中的值提供字符串或整数，则数据类型将不匹配。

相关术语

- 条件
- 条件数据块
- IAM JSON 策略元素：条件运算符
- 全局条件键
- Amazon 服务条件键

错误 — 无效的条件键格式

问题代码：INVALID_CONDITI_KEY_FORMAT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid condition key format: The condition key format is not valid. Use the format
 service:keyname.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key format is not valid. Use the format
 service:keyname."
```

解决错误

条件键值对中的键必须与服务的规格匹配。要查看每个服务定义的条件键，请参阅 Amazon 服务的操作、资源和条件键。请选择服务名称以查看该服务的条件键。

相关术语

- 条件
- 全局条件键
- Amazon 服务条件键

错误 — 无效条件多个布尔值

问题代码：INVALID_CONDION_MULTIPLE_BOLEAN

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid condition multiple Boolean: The condition key does not support multiple Boolean
 values. Use a single Boolean value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key does not support multiple Boolean values. Use a
 single Boolean value."
```

解决错误

条件键值对中的键需要一个布尔值。当您提供多个布尔值时，条件匹配可能不会返回您期望的结果。

要查看每个服务定义的条件键，请参阅 Amazon 服务的操作、资源和条件键。请选择服务名称以查看该服务的条件键。

- 条件

- 全局条件键
- Amazon 服务条件键

错误 — 无效的条件运算符

问题代码：INVALID_CONDITION_OPERATOR

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid condition operator: The condition operator {{operator}} is not valid. Use a
 valid condition operator.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition operator {{operator}} is not valid. Use a valid
 condition operator."
```

解决错误

请更新条件以使用受支持的条件运算符。

相关术语

- IAM JSON 策略元素：条件运算符
- 条件元素
- JSON 策略概述

错误 — 无效的效果

问题代码：INVALID_EFFECT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid effect: The effect {{effect}} is not valid. Use Allow or Deny.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The effect {{effect}} is not valid. Use Allow or Deny."
```

解决错误

请更新 `Effect` 元素以使用有效的效果。`Effect` 的有效值为 **`Allow`** 和 **`Deny`**。

相关术语

- 效果元素
- JSON 策略概述

错误 — 无效的全局条件键

问题代码：INVALID_GLOBAL_CONDION_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid global condition key: The condition key {{key}} does not exist. Use a valid
 condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{key}} does not exist. Use a valid condition
 key."
```

解决错误

请更新条件键值对中的条件键以使用受支持的全局条件键。

全局条件键是带有 `aws:` 前缀的条件键。Amazon 服务既可支持全局条件键，也可提供包含服务前缀的服务特定键。例如，IAM 条件键包含 `iam:` 前缀。有关更多信息，请参阅 Amazon 服务的操作、资源和条件键，然后选择要查看其键的服务。

相关术语

- 全局条件键

错误 — 无效的分区

问题代码：INVALID_PARTITION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid partition: The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN for the service {{service}} does not support the
 partition {{partition}}. Use the supported values: {{partitions}}"
```

解决错误

请更新资源 ARN 以包含支持的分区。如果您已包含支持的分区，则服务或资源可能不支持您包含的分区。

分区 是一组 Amazon 区域。每个 Amazon 账户的作用域为一个分区。在 Classic 区域中，使用 `aws` 分区。在中国区域中，使用 `aws-cn`。

相关术语

- Amazon Resource Names (ARN) - 分区

错误 — 无效的策略元素

问题代码：INVALID_POLICY_ELEMENT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid policy element: The policy element {{element}} is not valid.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy element {{element}} is not valid."
```

解决错误

请更新策略以仅包含受支持的 JSON 策略元素。

相关术语

- JSON 策略元素

错误 — 无效的主体格式

问题代码：INVALID_PRINCIPAL_FORMAT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid principal format: The Principal element contents are not valid. Specify a key-
value pair in the Principal element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element contents are not valid. Specify a key-value
 pair in the Principal element."
```

解决错误

更新主体以使用受支持的键值对格式。

您可以在基于资源的策略中指定主体，但不能采用基于身份的策略。

例如，要为 Amazon 账户中的每个人定义访问权限，请在策略中使用以下主体：

```
"Principal": { "AWS": "123456789012" }
```

相关术语

- JSON 策略元素：Principal
- 基于身份的策略和基于资源的策略

错误 — 无效的主体密钥

问题代码：INVALID_PRINCIPAL_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid principal key: The principal key {{principal-key}} is not valid.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The principal key {{principal-key}} is not valid."
```

解决错误

更新主体键值对中的键以使用受支持的主体密钥。以下是受支持的主体密钥：

- Amazon
- CanonicalUser
- 联合身份
- 服务

相关术语

- 主体要素

错误 — 无效的区域

问题代码：INVALID_REGION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid Region: The Region {{region}} is not valid. Update the condition value to a
 supported Region.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Region {{region}} is not valid. Update the condition value to a
 supported Region."
```

解决错误

请更新条件键值对的值以包含受支持的区域。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅区域列表。

相关术语

- 策略资源
- 资源 ARN
- 区域名称和代码

错误 — 无效的服务

问题代码：INVALID_SERVICE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid service: The service {{service}} does not exist. Use a valid service name.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service {{service}} does not exist. Use a valid service name."
```

解决错误

操作或条件键中的服务前缀必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅 Amazon 服务的操作、资源和条件键。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- 已知服务及其操作、资源和条件键

错误 — 无效的服务条件键

问题代码：INVALID_SERVICE_CONDION_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid service condition key: The condition key {{key}} does not exist in the service
 {{service}}. Use a valid condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{key}} does not exist in the service {{service}}.
 Use a valid condition key."
```

解决错误

请更新条件键值对中的键以使用服务的已知条件键。全局条件键名称以 aws 前缀开头。Amazon 服务可提供包含服务前缀的服务特定密钥。要查看服务的前缀，请参阅 Amazon 服务的操作、资源和条件键。

相关术语

- 全局条件键
- 已知服务及其操作、资源和条件键

错误 — 操作中的无效服务

问题代码：INVALID_SERVICE_IN_ACTION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid service in action: The service {{service}} specified in the action does not
 exist. Did you mean {{service2}}?
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service {{service}} specified in the action does not exist. Did
 you mean {{service2}}?"
```

解决错误

操作中的服务前缀必须与服务前缀的规范（包括大写）匹配。要查看服务的前缀，请参阅 Amazon 服务的操作、资源和条件键。请选择服务的名称并在第一句中找到服务的前缀。

相关术语

- 操作元素
- 已知服务及其行动

错误 — 无效的运算符的变量

问题代码：INVALID_VARIABLE_FOR_OPERATOR

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid variable for operator: Policy variables can only be used with String and ARN
 operators.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Policy variables can only be used with String and ARN operators."
```

解决错误

您可以在 Resource 元素中和 Condition 元素的字符串比较中使用策略变量。当您使用字符串运算符或 ARN 运算符时，条件支持变量。字符串运算符包括 StringEquals、StringLike 和 StringNotLike。ARN 运算符包括 ArnEquals 和 ArnLike。不能将策略变量与其他运算符（如数字、日期、布尔值、二进制、IP 地址或 Null 运算符）一起使用。

相关术语

- 在条件元素中使用策略变量
- 条件元素

错误 — 无效的版本

问题代码：INVALID_VERSION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid version: The version {{version}} is not valid. Use one of the following
 versions: {{versions}}
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The version {{version}} is not valid. Use one of the following
 versions: {{versions}}"
```

解决错误

Version 策略元素指定 Amazon 用于处理策略的语言语法规则。要使用所有可用策略功能，将以下 Version 元素包含在所有策略中的 Statement 元素之前。

```
"Version": "2012-10-17"
```

相关术语

- 版本元素

错误 — Json 语法错误

问题代码：JSON_SYNTAX_ERROR

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Json syntax error: Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Fix the JSON syntax error at index {{index}} line {{line}} column
 {{column}}."
```

解决错误

您的策略包含语法错误。请检查您的 JSON 语法。

相关术语

- JSON 验证程序
- IAM JSON 策略元素参考
- JSON 策略概述

错误 — Json 语法错误

问题代码：JSON_SYNTAX_ERROR

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Json syntax error: Fix the JSON syntax error.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Fix the JSON syntax error."
```

解决错误

您的策略包含语法错误。请检查您的 JSON 语法。

相关术语

- JSON 验证程序
- IAM JSON 策略元素参考
- JSON 策略概述

错误 — 缺少操作

问题代码：MISSING_ACTION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing action: Add an Action or NotAction element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add an Action or NotAction element to the policy statement."
```

解决错误

Amazon JSON 策略必须包含 `Action` 或者 `NotAction` 元素。

相关术语

- 操作元素
- NotAction 元素

- JSON 策略概述

错误 — 缺少 ARN 字段

问题代码：MISSING_ARN_FIELD

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing ARN field: Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Resource ARNs must include at least {{fields}} fields in the
 following structure: arn:partition:service:region:account:resource"
```

解决错误

资源 ARN 中的所有字段都必须与已知资源类型的规范相匹配。要查看服务的预期 ARN 格式，请参阅Amazon 服务的操作、资源和条件键。请选择服务的名称以查看其资源类型和 ARN 格式。

相关术语

- 策略资源
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 缺少 ARN 区域

问题代码：MISSING_ARN_REGION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing ARN Region: Add a Region to the {{service}} resource ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Region to the {{service}} resource ARN."
```

解决错误

大多数 Amazon 服务的资源 ARN 都需要您指定一个区域。有关每个区域（不带终端节点）支持的 Amazon 服务表，请参阅区域列表。

相关术语

- 策略资源
- 资源 ARN
- 区域名称和代码

错误 — 缺少效果

问题代码：MISSING_EFFECT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing effect: Add an Effect element to the policy statement with a value of Allow or
 Deny.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add an Effect element to the policy statement with a value of Allow
 or Deny."
```

解决错误

Amazon JSON 策略必须包含 `Effect` 元素，其值为 **`Allow`** 和 **`Deny`**。

相关术语

- 效果元素
- JSON 策略概述

错误 — 缺少主体

问题代码：MISSING_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing principal: Add a Principal element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Principal element to the policy statement."
```

解决错误

基于资源的策略必须包含 `Principal` 元素。

例如，要为 Amazon 账户中的每个人定义访问权限，请在策略中使用以下主体：

```
"Principal": { "AWS": "123456789012" }
```

相关术语

- 主体要素
- 基于身份的策略和基于资源的策略

错误 — 缺少限定符

问题代码：MISSING_QUALIFIER

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing qualifier: The request context key {{key}} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The request context key {{key}} has multiple values. Use the
 ForAllValues or ForAnyValue condition key qualifiers in your policy."
```

解决错误

在 Condition 元素中，您可构建表达式并使用条件运算符（等于、小于等）将策略中的条件键和值与请求上下文中的键和值进行匹配。对于包含单个条件键的多个值的请求，必须将条件括在方括号内，例如数组 ("Key2":["Value2A", "Value2B"])。您还必须将 ForAllValues 或 ForAnyValue 集合运算符与 StringLike 条件运算符搭配使用。这些限定词为条件运算符添加了集合运算功能，以便您针对多个条件值测试多个请求值。

相关术语

- 多值上下文键
- 条件元素

出现此错误的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其策略语句中包含缺少的条件键限定符。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您将 ForAllValues 或 ForAnyValue 条件键限定符添加到 Condition 元素。

- AWSGlueConsoleSageMakerNotebookFullAccess

错误 — 缺少资源

问题代码：MISSING_RESOURCE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing resource: Add a Resource or NotResource element to the policy statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a Resource or NotResource element to the policy statement."
```

解决错误

除角色信任策略之外的所有策略都必须包含 Resource 或 NotResource 元素。

相关术语

- 资源元素
- NotResource 元素
- 基于身份的策略和基于资源的策略
- JSON 策略概述

错误 — 缺少语句

问题代码：MISSING_STATEMENT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing statement: Add a statement to the policy
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a statement to the policy"
```

解决错误

JSON 策略必须包含语句。

相关术语

- JSON 策略元素

错误 — 如果存在则为空

问题代码：NULL_WITH_IF_EXISTS

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Null with if exists: The Null condition operator cannot be used with the IfExists
 suffix. Update the operator or the suffix.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Null condition operator cannot be used with the IfExists suffix.
 Update the operator or the suffix."
```

解决错误

除 `Null` 条件运算符外，您可在任何条件运算符名称的末尾添加 `IfExists`。使用 `Null` 条件运算符检查授权时是否有条件键。使用 `...ifExists` “如果请求的内容中存在策略键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 true。”

相关术语

- ...IfExists 条件运算符
- Null 条件运算符
- 条件元素

错误 — SCP 语法错误操作通配符

问题代码：SCP_SYNTAX_ERROR_ACTION_WILDCARD

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
SCP syntax error action wildcard: SCP actions can include wildcards (*) only at the end
 of a string. Update {{action}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "SCP actions can include wildcards (*) only at the end of a string.
 Update {{action}}."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 支持在 `Action` 或 `NotAction` 元素中指定值。但是，这些值只能在字符串末尾包含通配符 (*)。这意味着您可以指定 `iam:Get*` 但不能指定 `iam:*role`。

要指定多个操作，Amazon 建议您单独列出它们。

相关术语

- SCP 操作与 NotAction 元素
- SCP 评估
- Amazon Organizations 服务控制策略
- IAM JSON 策略元素：Action

错误 — SCP 语法错误主体

问题代码：SCP_SYNTAX_ERROR_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
SCP syntax error principal: SCPs do not support specifying principals. Remove the
 Principal or NotPrincipal element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "SCPs do not support specifying principals. Remove the Principal or
 NotPrincipal element."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 不支持 `Principal` 或 `NotPrincipal` 元素。

您可以使用 `aws:PrincipalArn` 全局条件键在 `Condition` 元素中指定 Amazon Resource Name (ARN)。

相关术语

- SCP 语法
- 主体的全局条件键

错误 — 需要唯一的 Sid

问题代码：UNIQUE_SIDS_REQUIRED

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unique Sids required: Duplicate statement IDs are not supported for this policy type.
 Update the Sid value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Duplicate statement IDs are not supported for this policy type.
 Update the Sid value."
```

解决错误

对于某些策略类型，语句 ID 必须是唯一的。`Sid`（声明 ID）元素允许您针对策略语句输入可选标识符。您可以在使用 `SID` 元素的语句数组中为每个语句指定语句 ID 的值。在允许您指定 ID 元素的产品中，例如 SQS 和 SNS，`Sid` 值正是策略文件 ID 的子 ID。例如，在 IAM 中，`Sid` 值必须在 JSON 策略中是唯一的。

相关术语

- IAM JSON 策略元素：Sid

错误 — 策略中不支持的操作

问题代码：UNSUPPORTED_ACTION_IN_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解决错误

附加到不同资源类型的基于资源策略中的 `Action` 元素不支持某些操作。例如，Amazon S3 存储桶策略中不支持 Amazon Key Management Service 操作。指定附加到基于资源的策略的资源类型支持的操作。

相关术语

- JSON 策略元素：Action

错误 — 不支持的元素组合

问题代码：UNSUPPORTED_ELEMENT_COMBINATION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported element combination: The policy elements {{element1}} and {{element2}} can
 not be used in the same statement. Remove one of these elements.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy elements {{element1}} and {{element2}} can not be used in
 the same statement. Remove one of these elements."
```

解决错误

某些 JSON 策略元素的组合不能一起使用。例如，您不能在同一策略语句中同时使用 `Action` 和 `NotAction`。相互排斥的其他对包括 `Principal/NotPrincipal` 和 `Resource/NotResource`。

相关术语

- IAM JSON 策略元素参考
- JSON 策略概述

错误 — 不支持的全局条件键

问题代码：UNSUPPORTED_GLOBAL_CONDITION_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported global condition key: The condition key aws:ARN is not supported. Use
 aws:PrincipalArn or aws:SourceArn instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key aws:ARN is not supported. Use aws:PrincipalArn or
 aws:SourceArn instead."
```

解决错误

Amazon 不支持使用指定的全局条件键。根据您的使用案例，您可以使用 `aws:PrincipalArn` 或 `aws:SourceArn` 全局条件键。例如，与 `aws:ARN` 不同，使用 `aws:PrincipalArn` 将发出请求的主体的 Amazon Resource Name (ARN) 与您在策略中指定的 ARN 进行比较。或者，使用 `aws:SourceArn` 全局条件键可将发出服务到服务请求的资源的 Amazon Resource Name (ARN) 与您在策略中指定的 ARN 进行比较。

相关术语

- Amazon 全局条件上下文密钥

错误 — 不支持的主体

问题代码：UNSUPPORTED_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported principal: The policy type {{policy_type}} does not support the Principal
 element. Remove the Principal element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy type {{policy_type}} does not support the Principal
 element. Remove the Principal element."
```

解决错误

`Principal` 元素指定允许或拒绝访问资源的主体。无法在基于 IAM 身份的策略中使用 `Principal` 元素。可以在 IAM 角色的信任策略和基于资源的策略中使用它。基于资源的策略是直接嵌入资源中的策略。例如，您可以在 Amazon S3 存储桶或 Amazon KMS 键中嵌入策略。

相关术语

- Amazon JSON 策略元素：Principal
- IAM 中的跨账户资源访问

错误 — 策略中不支持的资源 ARN

问题代码：UNSUPPORTED_RESOURCE_ARN_IN_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解决错误

策略附加到不同资源类型时，基于资源策略中的 `Resource` 元素不支持某些操作。例如，Amazon S3 存储桶策略的 `Resource` 元素中不支持 Amazon KMS ARN。指定附加到基于资源的策略的资源类型支持的资源 ARN。

相关术语

- JSON 策略元素：Action

错误 — 不支持的 Sid

问题代码：UNSUPPORTED_SID

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported Sid: Update the characters in the Sid element to use one of the following
 character types: [a-z, A-Z, 0-9]
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Update the characters in the Sid element to use one of the following
 character types: [a-z, A-Z, 0-9]"
```

解决错误

Sid 元素支持大写字母、小写字母和数字。

相关术语

- IAM JSON 策略元素：Sid

错误 — 主体中不支持的通配符

问题代码：UNSUPPORTED_WILDCARD_IN_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported wildcard in principal: Wildcards (*, ?) are not supported with the
 principal key {{principal_key}}. Replace the wildcard with a valid principal value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Wildcards (*, ?) are not supported with the principal key
 {{principal_key}}. Replace the wildcard with a valid principal value."
```

解决错误

Principal 元素结构支持使用键值对。策略中指定的主体值包含通配符 (*)。您不能在指定的主体密钥中包含通配符。当在 Principal 元素中指定用户时，不能使用通配符表示“所有用户”。您必须对一个特定用户或多个用户命名。同样地，当您指定代入角色会话时，不能使用通配符表示“所有会话”。您必须指明特定会话。同时，您不能使用通配符匹配一部分名称或 ARN。

要解决此结果，请移除通配符并提供更具体的主体。

相关术语

- Amazon JSON 策略元素：Principal

错误 — 变量中缺少大括号

问题代码：MISSING_BRACE_IN_VARIABLE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing brace in variable: The policy variable is missing a closing curly brace. Add }
 after the variable text.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy variable is missing a closing curly brace. Add } after
 the variable text."
```

解决错误

策略变量结构支持使用 $ 前缀，后跟一对大括号 ({ })。在 ${ } 字符内，包含想要在策略中使用的请求中的值名称。

若要解决此结果，请添加缺少的大括号，以确保存在完整的大括号开闭集。

相关术语

- IAM policy 元素：变量

错误 — 变量中缺少引号

问题代码：MISSING_QUOTE_IN_VARIABLE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing quote in variable: The policy variable default value must begin and end with a
 single quote. Add the missing quote.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy variable default value must begin and end with a single
 quote. Add the missing quote."
```

解决错误

向策略添加变量时，您可以为变量指定默认值。如果不存在变量，则 Amazon 使用您提供的原定设置文本。

要向变量添加默认值，请用单引号 (' ') 括起默认值，并用逗号和空格 (,) 分隔变量文本和默认值。

例如，如果主体使用 team=yellow 标记，他们可以访问名为 amzn-s3-demo-bucket-yellow 的 amzn-s3-demo-bucket Amazon S3 存储桶。使用此资源的策略可能允许团队成员访问自己的资源，但不能访问其他团队的资源。对于没有团队标签的用户，您可以将原定设置值设为 company-wide。这些用户只能访问 amzn-s3-demo-bucket-company-wide 存储桶，他们可以在其中查看广泛的信息，例如加入团队的说明。

```
"Resource":"arn:aws:s3:::amzn-s3-demo-bucket-${aws:PrincipalTag/team, 'company-wide'}"
```

相关术语

- IAM policy 元素：变量

错误 — 变量中不支持的空间

问题代码：UNSUPPORTED_SPACE_IN_VARIABLE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported space in variable: A space is not supported within the policy variable
 text. Remove the space.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "A space is not supported within the policy variable text. Remove the
 space."
```

解决错误

策略变量结构支持使用 $ 前缀，后跟一对大括号 ({ })。在 ${ } 字符内，包含想要在策略中使用的请求中的值名称。虽然在指定默认变量时可以包含空格，但不能在变量名称中包含空格。

相关术语

- IAM policy 元素：变量

错误 — 空变量

问题代码：EMPTY_VARIABLE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty variable: Empty policy variable. Remove the ${ } variable structure or provide a
 variable within the structure.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Empty policy variable. Remove the ${ } variable structure or provide
 a variable within the structure."
```

解决错误

策略变量结构支持使用 $ 前缀，后跟一对大括号 ({ })。在 ${ } 字符内，包含想要在策略中使用的请求中的值名称。

相关术语

- IAM policy 元素：变量

错误 — 元素中不支持变量

问题代码：VARIABLE_UNSUPPORTED_IN_ELEMENT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Variable unsupported in element: Policy variables are supported in the Resource and
 Condition elements. Remove the policy variable {{variable}} from this element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Policy variables are supported in the Resource and Condition
 elements. Remove the policy variable {{variable}} from this element."
```

解决错误

您可以在 `Resource` 元素中和 `Condition` 元素的字符串比较中使用策略变量。

相关术语

- IAM policy 元素：变量

错误 — 版本中不支持变量

问题代码：VARIABLE_UNSUPPORTED_IN_VERSION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Variable unsupported in version: To include variables in your policy, use the policy
 version 2012-10-17 or later.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "To include variables in your policy, use the policy version
 2012-10-17 or later."
```

解决错误

要使用策略变量，您必须在语句中包含 Version 元素，而且版本必须设置为支持策略变量的版本。变量是在 2012-10-17 版本中引入的。较早版本的策略语言不支持策略变量。如果您未将 Version 设置为 2012-10-17 或更高版本，则 ${aws:username} 等变量在策略中被视为文字字符串。

Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。当您更改 IAM 中的客户托管策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。

相关术语

- IAM policy 元素：变量
- IAM JSON 策略元素：Version

错误 — 私有 IP 地址

问题代码：PRIVATE_IP_ADDRESS

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Private IP address: aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "aws:SourceIp works only for public IP address ranges. The values
 for condition key aws:SourceIp include only private IP addresses and will not have the
 desired effect. Update the value to include only public IP addresses."
```

解决错误

全局条件键 aws:SourceIp 仅适用于公有 IP 地址范围。当您的策略仅允许私有 IP 地址时，您会收到此错误。在这种情况下，条件将永远不匹配。

- aws:SourceIp 全局条件键
- IAM JSON 策略元素：Condition

错误 — 私有 NotIpAddress

问题代码：PRIVATE_NOT_IP_ADDRESS

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Private NotIpAddress: The values for condition key aws:SourceIp include only private
 IP addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include only private IP
 addresses and has no effect. aws:SourceIp works only for public IP address ranges.
 Update the value to include only public IP addresses."
```

解决错误

全局条件键 `aws:SourceIp` 仅适用于公有 IP 地址范围。当您使用 `NotIpAddress` 条件运算符并仅列出私有 IP 地址时，就会收到此错误。在这种情况下，条件将始终匹配并且无效。

- aws:SourceIp 全局条件键
- IAM JSON 策略元素：Condition

错误 — 策略大小超过 SCP 配额

问题代码：POLICY_SIZE_EXCEDS_SCP_QUOTA

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Policy size exceeds SCP quota: The {{policySize}} characters in the service control
 policy (SCP) exceed the {{policySizeQuota}} character maximum for SCPs. We recommend
 that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{policySize}} characters in the service control policy (SCP)
 exceed the {{policySizeQuota}} character maximum for SCPs. We recommend that you use
 multiple granular policies."
```

解决错误

Amazon Organizations 服务控制策略 (SCP) 支持在 `Action` 或 `NotAction` 元素中指定值。但是，这些值只能在字符串末尾包含通配符 (*)。这意味着您可以指定 `iam:Get*` 但不能指定 `iam:*role`。

要指定多个操作，Amazon 建议您单独列出它们。

相关术语

- Amazon Organizations 的配额
- Amazon Organizations 服务控制策略

错误 — 无效的服务主体格式

问题代码：INVALID_SERVICE_PRINCIPAL_FORMAT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid service principal format: The service principal does not match the expected
 format. Use the format {{expectedFormat}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The service principal does not match the expected format. Use the
 format {{expectedFormat}}."
```

解决错误

条件键值对中的值必须与定义的服务主体格式匹配。

服务主体是一个标识符，用于向服务授予权限。您可以在 Principal 元素中指定服务主体，也可以将其作为某些全局条件键和特定于服务的键的值。服务主体由每项服务定义。

服务主体的标识符包括服务名称，通常采用以下格式（都是小写字母）：

service-name.amazonaws.com

某些特定于服务的键可能会对服务主体使用不同的格式。例如，kms:ViaService 条件密钥要求所有小写字母的服务主体采用以下格式：

service-name.AWS_region.amazonaws.com

相关术语

- 服务主体
- Amazon 全局条件键
- kms:ViaService 条件键

错误 — 条件中缺少标签密钥

问题代码：MISSING_TAG_KEY_IN_CONDITION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing tag key in condition: The condition key {{conditionKeyName}} must include a tag
 key to control access based on tags. Use the format {{conditionKeyName}}tag-key and
 specify a key name for tag-key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{conditionKeyName}} must include a tag key to
 control access based on tags. Use the format {{conditionKeyName}}tag-key and specify a
 key name for tag-key."
```

解决错误

要基于标签控制访问，您需要在策略的条件元素中提供标签信息。

例如，要控制对 Amazon 资源的访问权限，您可以包含 aws:ResourceTag 条件键。此键需要格式 aws:ResourceTag/*tag-key*。要指定条件中的标签密钥 owner 和标签值 JaneDoe，请使用以下格式。

```
"Condition": {
    "StringEquals": {"aws:ResourceTag/owner": "JaneDoe"}
}
```

相关术语

- 使用标签控制访问
- 条件
- 全局条件键
- Amazon 服务条件键

错误 — VPC 格式无效

问题代码：INVALID_VPC_FORMAT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid vpc format: The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The VPC identifier in the condition key value is not valid. Use the
 prefix 'vpc-' followed by 8 or 17 alphanumeric characters."
```

解决错误

aws:SourceVpc 条件密钥必须使用前缀 vpc-，并且后面跟 8 或 17 个字母数字字符，例如 vpc-11223344556677889 或 vpc-12345678。

相关术语

- Amazon 全局条件密钥：aws:SourceVpc

错误 — vpce 格式无效

问题代码：INVALID_VPCE_FORMAT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid vpce format: The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The VPCE identifier in the condition key value is not valid.  Use
 the prefix 'vpce-' followed by 8 or 17 alphanumeric characters."
```

解决错误

aws:SourceVpce 条件密钥必须使用前缀 vpce-，并且后面跟 8 或 17 个字母数字字符，例如 vpce-11223344556677889 或 vpce-12345678。

相关术语

- Amazon 全局条件密钥：aws:SourceVpce

错误 — 不支持联合主体

问题代码：FEDERATED_PRINCIPAL_NOT_SUPPORTED

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Federated principal not supported: The policy type does not support a federated
 identity provider in the principal element. Use a supported principal.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The policy type does not support a federated identity provider in
 the principal element. Use a supported principal."
```

解决错误

这些区域有：Principal 元素将联合主体用于附加到 IAM 角色的信任策略，以便通过联合身份验证提供访问权限 标识策略和其他基于资源的策略不支持 Principal 元素中的联合标识提供程序。例如，您不能在 Amazon S3 存储桶策略中使用 SAML 主体。将 Principal元素更改为受支持的主体类型。

相关术语

- 创建适用于身份联合验证的角色
- JSON 策略元素：Principal

错误 — 条件密钥的不支持的操作

问题代码：UNSUPPORTED_ACTION_FOR_CONDITION_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported action for condition key: The following actions: {{actions}} are not
 supported by the condition key {{key}}. The condition will not be evaluated for these
 actions. We recommend that you move these actions to a different statement without
 this condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The following actions: {{actions}} are not supported by the
 condition key {{key}}. The condition will not be evaluated for these actions. We
 recommend that you move these actions to a different statement without this condition
 key."
```

解决错误

确保策略声明 Condition 元素中的条件密钥适用于 Action 元素中的每个操作。为确保您的策略有效允许或拒绝指定的操作，应将不受支持的操作移动到不带条件密钥的其他语句中。

> **Note**
>
> 如果 Action 元素具有带通配符的操作，IAM Access Analyzer 不会对此错误的这些操作进行评估。

相关术语

- JSON 策略元素：Action

错误 — 策略中不支持的操作

问题代码：UNSUPPORTED_ACTION_IN_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported action in policy: The action {{action}} is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The action {{action}} is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解决错误

附加到不同资源类型的基于资源策略中的 Action 元素不支持某些操作。例如，Amazon S3 存储桶策略中不支持 Amazon Key Management Service 操作。指定附加到基于资源的策略的资源类型支持的操作。

相关术语

- JSON 策略元素：Action

错误 — 策略中不支持的资源 ARN

问题代码：UNSUPPORTED_RESOURCE_ARN_IN_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported resource ARN in policy: The resource ARN is not supported for the resource-
based policy attached to the resource type {{resourceType}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource ARN is not supported for the resource-based policy
 attached to the resource type {{resourceType}}."
```

解决错误

策略附加到不同资源类型时，基于资源策略中的 `Resource` 元素不支持某些操作。例如，Amazon S3 存储桶策略的 `Resource` 元素中不支持 Amazon KMS ARN。指定附加到基于资源的策略的资源类型支持的资源 ARN。

相关术语

- JSON 策略元素：Action

错误 — 服务主体的条件密钥不受支持

问题代码：UNSUPPORTED_CONDITION_KEY_FOR_SERVICE_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unsupported condition key for service principal: The following condition keys are not
 supported when used with the service principal: {{conditionKeys}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The following condition keys are not supported when used with the
 service principal: {{conditionKeys}}."
```

解决错误

您可以使用服务主体指定基于资源策略的 `Principal` 元素中的 Amazon Web Services 服务，这是服务的标识符。您不能对某些服务主体使用某些条件密钥。例如，您不能使用带有服务主体 `cloudfront.amazonaws.com` 的 `aws:PrincipalOrgID` 条件密钥。您应该移除 `Principal` 元素中不适用于服务主体的条件密钥。

相关术语

- 服务主体
- JSON 策略元素：Principal

错误 - 角色信任策略语法错误 notprincipal

问题代码：OLE_TRUST_POLICY_SYNTAX_ERROR_NOTPRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Role trust policy syntax error notprincipal: Role trust policies do not support
 NotPrincipal. Update the policy to use a Principal element instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Role trust policies do not support NotPrincipal. Update the policy
 to use a Principal element instead."
```

解决错误

角色信任策略是附加到 IAM 角色的、基于资源的策略。信任策略定义哪些主体实体（账户、用户、角色和联合身份用户）可以代入该角色。角色信任策略不支持 `NotPrincipal`。更新策略以改用 `Principal` 元素。

相关术语

- JSON 策略元素：Principal
- JSON 策略元素：NotPrincipal

错误 - 主体中不支持的角色信任策略通配符

问题代码：ROLE_TRUST_POLICY_UNSUPPORTED_WILDCARD_IN_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Role trust policy unsupported wildcard in principal: "Principal:" "*" is not supported
 in the principal element of a role trust policy. Replace the wildcard with a valid
 principal value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": ""Principal:" "*" is not supported in the principal element of a role
 trust policy. Replace the wildcard with a valid principal value."
```

解决错误

角色信任策略是附加到 IAM 角色的、基于资源的策略。信任策略定义哪些主体实体（账户、用户、角色和联合用户）可以代入该角色。角色信任策略的 `Principal` 元素中不支持 `"Principal:" "*"`。将通配符替换为有效的主体值。

相关术语

- JSON 策略元素：Principal

错误 - 角色信任策略语法错误资源

问题代码：ROLE_TRUST_POLICY_SYNTAX_ERROR_RESOURCE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Role trust policy syntax error resource: Role trust policies apply to the role
 that they are attached to. You cannot specify a resource. Remove the Resource or
 NotResource element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Role trust policies apply to the role that they are attached to. You
 cannot specify a resource. Remove the Resource or NotResource element."
```

解决错误

角色信任策略是附加到 IAM 角色的、基于资源的策略。信任策略定义哪些主体实体（账户、用户、角色和联合身份用户）可以代入该角色。角色信任策略适用于其附加到的角色。您不能在角色信任策略中指定 `Resource` 或 `NotResource` 元素。删除 `Resource` 或 `NotResource` 元素。

- JSON 策略元素：Resource
- JSON 策略元素：NotResource

错误 - 类型与 IP 范围不匹配

问题代码：TYPE_MISMATCH_IP_RANGE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch IP range: The condition operator {{operator}} is used with an invalid IP
 range value. Specify the IP range in standard CIDR format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition operator {{operator}} is used with an invalid IP range
 value. Specify the IP range in standard CIDR format."
```

解决错误

请更新文本以使用 CIDR 格式的 IP 地址条件运算符数据类型。

相关术语

- IP 地址条件运算符
- IAM JSON 策略元素：条件运算符

错误 - 缺少条件键的操作

问题代码：MISSING_ACTION_FOR_CONDITION_KEY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing action for condition key: The {{actionName}} action must be in the action block
 to allow setting values for the condition key {{keyName}}. Add {{actionName}} to the
 action block.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{actionName}} action must be in the action block to allow
 setting values for the condition key {{keyName}}. Add {{actionName}} to the action
 block."
```

解决错误

除非指定的操作在 `Action` 元素中，否则不会评估策略语句 `Condition` 元素中的条件键。要确保策略有效地允许或拒绝您指定的条件键，请将操作添加到 `Action` 元素。

相关术语

- JSON 策略元素：Action

错误 - 角色信任策略中的联合主体语法无效

问题代码：INVALID_FEDERATED_PRINCIPAL_SYNTAX_IN_ROLE_TRUST_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid federated principal syntax in role trust policy: The principal value specifies
 a federated principal that does not match the expected format. Update the federated
 principal to a domain name or a SAML metadata ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The principal value specifies a federated principal that does not
 match the expected format. Update the federated principal to a domain name or a SAML
 metadata ARN."
```

解决错误

主体值指定的联合主体与预期格式不匹配。将联合主体的格式更新为有效的域名或 SAML 元数据 ARN。

相关术语

- 联合用户和角色

错误 - 主体的操作不匹配

问题代码：MISMATCHED_ACTION_FOR_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Mismatched action for principal: The {{actionName}} action is invalid with the
 following principal(s): {{principalNames}}. Use a SAML provider principal with
 the sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{actionName}} action is invalid with the following
 principal(s): {{principalNames}}. Use a SAML provider principal with the
 sts:AssumeRoleWithSAML action or use an OIDC provider principal with the
 sts:AssumeRoleWithWebIdentity action. Ensure the provider is Federated if you use
 either of the two options."
```

解决错误

策略语句 `Action` 元素中指定的操作对于 `Principal` 元素中指定的主体无效。例如，不能将 SAML 提供商主体与 `sts:AssumeRoleWithWebIdentity` 操作一起使用。应将 SAML 提供商主体与 `sts:AssumeRoleWithSAML` 操作一起使用，或者将 OIDC 提供商主体与 `sts:AssumeRoleWithWebIdentity` 操作一起使用。

相关术语

- AssumeRoleWithSAML
- AssumeRoleWithWebIdentity

错误 - 缺少 Roles Anywhere 信任策略的操作

问题代码：MISSING_ACTION_FOR_ROLES_ANYWHERE_TRUST_POLICY

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing action for roles anywhere trust policy: The rolesanywhere.amazonaws.com service
 principal requires the sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession
 permissions to assume a role. Add the missing permissions to the policy.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The rolesanywhere.amazonaws.com service principal requires the
 sts:AssumeRole, sts:SetSourceIdentity, and sts:TagSession permissions to assume a
 role. Add the missing permissions to the policy."
```

解决错误

要使 IAM Roles Anywhere 能够代入角色并提供临时 Amazon 凭证，该角色必须信任 IAM Roles Anywhere 服务主体。IAM Roles Anywhere 服务主体需要 `sts:AssumeRole`、`sts:SetSourceIdentity` 和 `sts:TagSession` 权限才能代入角色。如果缺少任何权限，则必须将其添加到策略中。

相关术语

- Amazon Identity and Access Management Roles Anywhere 中的信任模式

错误 – 策略大小超过 RCP 配额

问题代码：POLICY_SIZE_EXCEDS_RCP_QUOTA

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Policy size exceeds RCP quota: The {{policySize}} characters in the resource control
 policy (RCP) exceed the {{policySizeQuota}} character maximum for RCPs. We recommend
 that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{policySize}} characters in the resource control policy (RCP)
 exceed the {{policySizeQuota}} character maximum for RCPs. We recommend that you use
 multiple granular policies."
```

解决错误

Amazon Organizations 资源控制策略（RCP）支持在 `Action` 元素中指定值。但是，这些值只能在字符串末尾包含通配符 (*)。这意味着您可以指定 `s3:Get*` 但不能指定 `s3:*Object`。

要指定多个操作，Amazon 建议您单独列出它们。

相关术语

- Amazon Organizations 的配额

- Amazon Organizations 资源控制策略

错误 – RCP 语法错误主体

问题代码：RCP_SYNTAX_ERROR_PRINCIPAL

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP syntax error principal: The Principal element contents are not valid. RCPs only
 support specifying all principals ("*") in the Principal element. The NotPrincipal
 element is not supported for RCPs.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element contents are not valid. RCPs only support
 specifying all principals ("*") in the Principal element. The NotPrincipal element is
 not supported for RCPs."
```

解决错误

Amazon Organizations 资源控制策略（RCP）仅支持在 `Principal` 元素中指定所有主体（“*”）。`NotPrincipal` 元素不支持 RCP。

相关术语

- RCP 语法
- 主体的属性

错误 – RCP 语法错误允许

问题代码：RCP_SYNTAX_ERROR_ALLOW

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP syntax error allow: RCPs only support specifying all principals ("*") in the
 Principal element, all resources ("*") in the Resource element, and no Condition
 element with an effect of Allow.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "RCPs only support specifying all principals ("*") in the Principal
 element, all resources ("*") in the Resource element, and no Condition element with an
 effect of Allow."
```

解决错误

Amazon Organizations 资源控制策略（RCP）仅支持在 `Principal` 元素中指定所有主体（“*”），以及在 `Resource` 元素中指定所有资源（“*”）。效果为 `Allow` 的 `Condition` 元素不支持 RCP。

相关术语

- RCP 语法
- 主体的属性
- 资源的属性

错误 – RCP 语法错误 NotAction

问题代码：RCP_SYNTAX_ERROR_NOTACTION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP syntax error NotAction: RCPs do not support the NotAction element. Update to use
 the Action element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "RCPs do not support the NotAction element. Update to use the Action
 element."
```

解决错误

Amazon Organizations 资源控制策略（RCP）不支持 `NotAction` 元素。使用 `Action` 元素。

相关术语

- RCP 语法
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：NotAction

错误 – RCP 语法错误操作

问题代码：RCP_SYNTAX_ERROR_ACTION

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP syntax error action: RCPs only support specifying select service prefixes in the
 Action element. Learn more here.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "RCPs only support specifying select service prefixes in the Action
 element. Learn more here."
```

解决错误

Amazon Organizations 资源控制策略（RCP）仅支持在 `Action` 元素中指定 Select 服务前缀。

相关术语

- RCP 语法
- 支持 RCP 的 Amazon Web Services 服务 列表

错误 - 缺少 ARN 账户

问题代码：MISSING_ARN_ACCOUNT

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing ARN account: The resource {{resourceName}} in the arn is missing an account id.
 Please provide a 12 digit account id.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The resource {{resourceName}} in the arn is missing an account id.
 Please provide a 12 digit account id."
```

解决错误

在资源 ARN 中包含账户 ID。账户 ID 是 12 位整数。要了解如何查看您的账户 ID，请参阅查找您的 Amazon 账户 ID。

相关术语

- 策略资源
- 账户标识符
- 资源 ARN
- 采用 ARN 格式的 Amazon 服务资源

错误 — 无效的 kms 密钥值

问题代码：INVALID_KMS_KEY_VALUE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid kms key value: The {{key}} condition key value must be valid a KMS key ARN.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{key}} condition key value must be valid a KMS key ARN."
```

解决错误

Amazon KMS key ARN（Amazon 资源名称）是 KMS 密钥唯一的完全限定标识符。密钥 ARN 包括 Amazon Web Services 账户、区域和密钥 ID。密钥 ARN 遵循以下格式：

`arn:aws:kms:region:account-id:key/key-id`

相关术语

- 查找密钥 ID 和密钥 ARN
- 密钥 ARN
- Amazon 全局条件上下文密钥

错误 – 变量使用过于宽容

问题代码：VARIABLE_USAGE_TOO_PERMISSIVE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Variable usage too permissive: Overly permissive use of policy variable for the {{key}}
 condition key. Use the policy variable preceded by 6 consecutive characters.
```

```
The policy variable is not allowed in the condition key {{key}}. We consider the key to
 be sensitive and policy variables can be evaluated as effective wildcards. Therefore
 policy variables are not allowed to be used with sensitive keys. Refer to https://
docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-keys.html for the
 list of sensitive keys.
```

```
Overly permissive use of policy variable with aws:userID. Use the policy variable on
 the right side of colon or as the only character on the left side of a colon.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Overly permissive use of policy variable for the {{key}} condition
 key. Use the policy variable preceded by 6 consecutive characters."
```

```
"findingDetails": "The policy variable is not allowed in the condition key {{key}}.
 We consider the key to be sensitive and policy variables can be evaluated as
 effective wildcards. Therefore policy variables are not allowed to be used
 with sensitive keys. Refer to https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_condition-keys.html for the list of sensitive keys."
```

```
"findingDetails": "Overly permissive use of policy variable with aws:userID. Use the
 policy variable on the right side of colon or as the only character on the left side
 of a colon."
```

解决错误

此错误有三条可能的调查发现消息。

- 对于第一条错误消息，请修改您的策略变量用法，使其更具针对性。在策略变量前添加至少 6 个连续字符，以缩小权限范围。例如，使用 `prefix-${aws:username}` 或 `myapp-${aws:username}`，而不使用 `${aws:username}`。这样可以确保该策略变量不会授予过于广泛的访问权限。
- 对于第二条错误消息，请从指定的条件键中移除策略变量。策略变量可以起到有效的通配符作用，出于安全考虑，不允许将其与敏感条件键一同使用。相反，请使用特定的静态值，或者考虑重新设计您的策略，以使用支持策略变量的非敏感条件键。
- 对于第三条错误消息，请修改您的 `aws:userID` 策略变量用法，使其更加严格。将策略变量置于冒号右侧（在账户 ID 之后）或者将其用作冒号左侧的唯一字符。例如，使用 `AIDACKCEVSQ6C2EXAMPLE:${aws:userid}` 或 `${aws:userid}:*` 而不是 `${aws:userid}`。

相关术语

- IAM 策略元素：变量和标签
- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥

错误 – 通配符使用过于宽容

问题代码：WILDCARD_USAGE_TOO_PERMISSIVE

调查发现类型：ERROR

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Wildcard usage too permissive: Overly permissive use of wildcard for the {{key}}
 condition key. Use the wildcard preceded by 6 consecutive characters.
```

```
The wildcard is not allowed in the condition key {{key}}. We consider the key to
 be sensitive and wildcards are not allowed to be used with sensitive keys. Refer
 to https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_condition-
keys.html for the list of sensitive keys.
```

```
Overly permissive use of wildcard with aws:userID. Use the wildcard on the right side
 of colon or as the only character on the left side of a colon.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Overly permissive use of wildcard for the {{key}} condition key. Use
 the wildcard preceded by 6 consecutive characters."
```

```
"findingDetails": "The wildcard is not allowed in the condition key {{key}}.
 We consider the key to be sensitive and wildcards are not allowed to be used
 with sensitive keys. Refer to https://docs.aws.amazon.com/IAM/latest/UserGuide/
reference_policies_condition-keys.html for the list of sensitive keys."
```

```
"findingDetails": "Overly permissive use of wildcard with aws:userID. Use the wildcard
 on the right side of colon or as the only character on the left side of a colon."
```

解决错误

此错误有三条可能的调查发现消息。

- 对于第一条错误消息，请在通配符前添加至少 6 个连续字符，使通配符的使用更具针对性。例如，使用 `prefix-*` 或 `prefix-*-suffix`，而不使用 `*`。这缩小了该条件的适用范围，并遵循了最低权限原则。
- 对于第二条错误消息，请从指定的条件键中移除通配符。出于安全考虑，不允许将通配符与敏感条件键一同使用。将通配符替换为与您预期的访问模式相匹配的特定值，或者考虑使用支持通配符的其他非敏感条件键。

- 对于第三条错误消息，请修改您的 `aws:userID` 通配符用法，使其更加严格。将通配符置于冒号右侧（在账户 ID 之后）或者将其用作冒号左侧的唯一字符。例如，使用 `AIDACKCEVSQ6C2EXAMPLE:*` 或 `*:*` 而不是 `*`。

相关术语

- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥
- IAM 标识符

常规警告 — 使用 NotResource 创建 SLR

问题代码：CREATE_SLR_WITH_NOT_RESOURCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with NotResource: Using the iam:CreateServiceLinkedRole action with
 NotResource can allow creation of unintended service-linked roles for multiple
 resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with NotResource can
 allow creation of unintended service-linked roles for multiple resources. We recommend
 that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有 `NotResource` 元素的策略中使用 `iam:CreateServiceLinkedRole` 可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：NotResource

- IAM JSON 策略元素：Resource

常规警告 — 使用带有星号的操作和 NotResource 创建 SLR

问题代码：CREATE_SLR_WITH_STAR_IN_ACTION_AND_NOT_RESOURCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with star in action and NotResource: Using an action with a wildcard(*) and
 NotResource can allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using an action with a wildcard(*) and NotResource can
 allow creation of unintended service-linked roles because it can allow
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在 `Action` 中具有通配符 (*) 且 `NotResource` 元素的策略可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Resource

常规警告 — 使用 NotAction 和 NotResource 创建 SLR

问题代码：CREATE_SLR_WITH_NOT_ACTION_AND_NOT_RESOURCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with NotAction and NotResource: Using NotAction with NotResource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on multiple resources. We recommend that you
 specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using NotAction with NotResource can allow creation of unintended
 service-linked roles because it allows iam:CreateServiceLinkedRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。将 `NotAction` 元素与 `NotResource` 元素搭配使用可以允许为多个资源创建意外的服务链接角色。Amazon 会建议您重写策略以允许 `Resource` 元素中 ARN 有限列表上的 `iam:CreateServiceLinkedRole`。您还可以将 `iam:CreateServiceLinkedRole` 添加到 `NotAction` 元素。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：NotAction
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Resource

常规警告 — 使用带星号的资源创建 SLR

问题代码：CREATE_SLR_WITH_STAR_IN_RESOURCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with star in resource: Using the iam:CreateServiceLinkedRole action with
 wildcards (*) in the resource can allow creation of unintended service-linked roles.
 We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:CreateServiceLinkedRole action with wildcards (*) in
 the resource can allow creation of unintended service-linked roles. We recommend that
 you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有包含通配符 (*) 的 `Resource` 元素的策略中使用 `iam:CreateServiceLinkedRole` 可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：Resource

带有此常规警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您的账户中的高级用户。以下 Amazon 托管策略提供高级用户访问权限并授予为任何 Amazon 服务创建服务关联角色的权限。Amazon 建议您仅将以下 Amazon 托管策略附加到您认为是高级用户的 IAM 身份。

- PowerUserAccess
- AlexaForBusinessFullAccess
- AWSOrganizationsServiceTrustPolicy — 此 Amazon 托管策略提供了权限以供 Amazon Organizations 服务关联角色使用。此角色允许 Organizations 为 Amazon 企业中的其他服务创建额外的服务关联角色。

常规警告 — 使用带星号的行动和资源创建 SLR

问题代码：CREATE_SLR_WITH_STAR_IN_ACTION_AND_RESOURCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with star in action and resource: Using wildcards (*) in the action and
 the resource can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using wildcards (*) in the action and the resource
 can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在 `Action` 和 `Resource` 元素中带有通配符 (*) 的策略可以允许为多个资源创建意外的服务关联角色。这允许您在指定 `"Action": "*"`、`"Action": "iam:*"` 或 `"Action": "iam:Create*"` 时创建服务关联角色。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：Resource

带有此常规警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限并授予为任何 Amazon 服务创建服务关联角色的权限。Amazon 建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- AdministratorAccess
- IAMFullAccess

常规警告 — 使用带有星号的资源和 NotAction 创建 SLR

问题代码：CREATE_SLR_WITH_STAR_IN_RESOURCE_AND_NOT_ACTION

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Create SLR with star in resource and NotAction: Using a resource with wildcards (*)
 and NotAction can allow creation of unintended service-linked roles because it allows
 iam:CreateServiceLinkedRole permissions on all resources. We recommend that you
 specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can allow creation
 of unintended service-linked roles because it allows iam:CreateServiceLinkedRole
 permissions on all resources. We recommend that you specify resource ARNs instead."
```

解决一般警告

操作 `iam:CreateServiceLinkedRole` 授予权限以创建 IAM 角色，它允许 Amazon 服务代表您执行操作。在带有包含通配符 (*) 的 `Resource` 元素的策略中使用 `NotAction` 元素可以允许为多个资源创建意外的服务关联角色。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。您还可以将 `iam:CreateServiceLinkedRole` 添加到 `NotAction` 元素。

- CreateServiceLinkedRole 操作
- IAM JSON 策略元素：NotAction
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：Resource

常规警告 — 已弃用的全局条件键

问题代码：DEPRECATED_GLOBAL_CONDITION_KEY

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Deprecated global condition key: We recommend that you update aws:ARN to use the newer
 condition key aws:PrincipalArn.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you update aws:ARN to use the newer condition key
 aws:PrincipalArn."
```

解决一般警告

该策略包含已弃用的全局条件键。请更新条件键值对中的条件键以使用受支持的全局条件键。

- 全局条件键

常规警告 — 无效的日期值

问题代码：INVALID_DATE_VALUE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid date value: The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The date {{date}} might not resolve as expected. We recommend that
 you use the YYYY-MM-DD format."
```

解决一般警告

Unix Epoch 时间描述自 1970 年 1 月 1 日以来已经过去的时间点，减去闰秒。Epoch 时间可能无法解析到您期望的精确时间。Amazon 建议您对日期和时间格式使用 W3C 标准。例如，您可以指定一个完整的日期，如 YYYY-MMM-DD (1997-07-16)，也可以将时间附加到秒，例如 YYYY-MM-DDThh:mm:ssTZD (1997-07-16T19:20:30+01:00)。

- W3C 日期和时间格式
- IAM JSON 策略元素：Version
- aws:CurrentTime 全局条件键

常规警告 — 无效的角色引用

问题代码：INVALID_ROLE_REFERENCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid role reference: The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element includes the IAM role ID {{roleid}}. We
 recommend that you use a role ARN instead."
```

解决一般警告

Amazon 建议您为 IAM 角色指定 Amazon Resource Name (ARN)，而不是其主体 ID。当 IAM 保存策略时，它会将 ARN 转换为现有角色的主体 ID。Amazon 包括安全预防措施。如果某人删除并重新创建该角色，则该角色将有一个新 ID，并且策略不会与新角色的 ID 匹配。

- 指定主体：IAM 角色
- IAM ARN
- IAM 唯一 ID

常规警告 — 无效的用户引用

问题代码：INVALID_USER_REFERENCE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Invalid user reference: The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The Principal element includes the IAM user ID {{userid}}. We
 recommend that you use a user ARN instead."
```

解决一般警告

Amazon 建议您为 IAM 用户指定 Amazon Resource Name (ARN)，而不是其主体 ID。当 IAM 保存策略时，它会将 ARN 转换为现有用户的主体 ID。Amazon 包括安全预防措施。如果某人删除并重新创建该用户，则该用户将有一个新 ID，并且策略不会与新用户的 ID 匹配。

- 指定主体：IAM 用户
- IAM ARN
- IAM 唯一 ID

常规警告 — 缺少版本

问题代码：MISSING_VERSION

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing version: We recommend that you specify the Version element to help you with
 debugging permission issues.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you specify the Version element to help you with
 debugging permission issues."
```

解决一般警告

Amazon 建议在您的策略中包括可选的 `Version` 参数。如果未包含 Version 元素，则此值默认为 `2012-10-17`，但策略变量等较新的功能将不适用于您的策略。例如，系统不会将 `${aws:username}` 等变量识别为变量，而是将其视为策略中的文本字符串。

- IAM JSON 策略元素：Version

一般警告 — 建议的使用唯一的 Sid

问题代码：UNIQUE_SIDS_RECOMMENDED

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Unique Sids recommended: We recommend that you use statement IDs that are unique to
 your policy. Update the Sid value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend that you use statement IDs that are unique to your
 policy. Update the Sid value."
```

解决一般警告

Amazon 建议您使用唯一语句 ID。`Sid` (声明 ID) 元素允许您针对策略语句输入可选标识符。您可以在使用 `SID` 元素的语句数组中为每个语句指定语句 ID 的值。

相关术语

- IAM JSON 策略元素：Sid

常规警告 — 没有类似运算符的通配符

问题代码：WILDCARD_WITHOUT_LIKE_OPERATOR

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Wildcard without like operator: Your condition value includes a * or ? character. If
 you meant to use a wildcard (*, ?), update the condition operator to include Like.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Your condition value includes a * or ? character. If you meant to
 use a wildcard (*, ?), update the condition operator to include Like."
```

解决一般警告

Condition 元素结构要求您使用条件运算符和键值对。当您指定使用通配符 (*, ?) 的条件值时，必须使用 Like 版本的条件运算符。例如，请使用 StringLike，而不是 StringEquals 字符串条件运算符。

```
"Condition": {"StringLike": {"aws:PrincipalTag/job-category": "admin-*"}}
```

- IAM JSON 策略元素：条件运算符
- IAM JSON 策略元素：Condition

带有此常规警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略在其条件值中包含通配符，而不带有包含用于模式匹配的 Like 的条件运算符。使用 Amazon 托管策略作为创建客户管理策略的参考时，Amazon 会建议您使用支持模式匹配且带有通配符 (*, ?) 的条件运算符，例如 StringLike。

- AWSGlueConsoleSageMakerNotebookFullAccess

常规警告 — 策略大小超出身份策略配额

问题代码：POLICY_SIZE_EXCEEDS_IDENTITY_POLICY_QUOTA

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Policy size exceeds identity policy quota: The {{policySize}} characters in the
 identity policy, excluding whitespace, exceed the {{policySizeQuota}} character
 maximum for inline and managed policies. We recommend that you use multiple granular
 policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{policySize}} characters in the identity policy, excluding
 whitespace, exceed the {{policySizeQuota}} character maximum for inline and managed
 policies. We recommend that you use multiple granular policies."
```

解决一般警告

您可以将最多 10 个托管策略附加到 IAM 身份（用户、用户组或角色）。但是，每个托管策略的大小不能超过默认配额的 6144 个字符。在计算策略大小时，IAM 不会将空格计入该配额。配额，也称为 Amazon 中的限制，是 Amazon 账户中资源、操作和项目的最大值。

此外，您还可以向 IAM 身份添加所需数量的内联策略。但是，每个身份的所有内联策略的总大小不能超过指定的配额。

如果您的策略大于配额，您可以将策略组织为多个语句，并将这些语句分组为多个策略。

相关术语

- IAM 和Amazon STS STS 字符配额
- 多个声明和多个策略
- IAM 客户托管策略
- JSON 策略概述
- IAM JSON 策略语法

带有此常规警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

以下 Amazon 托管策略向跨多个 Amazon 服务的操作授予权限，且超过了最大策略大小。使用 Amazon 托管策略作为参考来创建托管策略时，您必须将策略拆分为多个策略。

- ReadOnlyAccess
- AWSSupportServiceRolePolicy

常规警告 — 策略大小超出资源策略配额

问题代码：POLICY_SIZE_EXCEEDS_RESOURCE_POLICY_QUOTA

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Policy size exceeds resource policy quota: The {{policySize}} characters in the
 resource policy exceed the {{policySizeQuota}} character maximum for resource
 policies. We recommend that you use multiple granular policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{policySize}} characters in the resource policy exceed the
 {{policySizeQuota}} character maximum for resource policies. We recommend that you use
 multiple granular policies."
```

解决一般警告

基于资源的策略是附加到资源（如 Amazon S3 存储桶）的 JSON 策略文档。这些策略授予指定的主体对该资源执行特定操作的权限，并定义这在哪些条件下适用。基于资源的策略的大小不能超过为该资源设置的配额。配额，也称为 Amazon 中的限制，是 Amazon 账户中资源、操作和项目的最大值。

如果您的策略大于配额，您可以将策略组织为多个语句，并将这些语句分组为多个策略。

相关术语

- 基于资源的策略
- Amazon S3 存储桶策略
- 多个声明和多个策略
- JSON 策略概述
- IAM JSON 策略语法

常规警告 — 类型不匹配

问题代码：TYPE_MISMATCH

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch: Use the operator type {{allowed}} instead of operator {{operator}} for
 the condition key {{key}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Use the operator type {{allowed}} instead of operator {{operator}}
 for the condition key {{key}}."
```

解决一般警告

请更新文本以使用支持的条件运算符数据类型。

例如，aws:MultiFactorAuthPresent 全局条件键需要带有 Boolean 数据类型的条件运算符。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- 全局条件键
- IAM JSON 策略元素：条件运算符

常规警告 — 类型不匹配布尔值

问题代码：TYPE_MISMATCH_BOLEAN

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch Boolean: Add a valid Boolean value (true or false) for the condition
 operator {{operator}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid Boolean value (true or false) for the condition operator
 {{operator}}."
```

解决一般警告

请更新文本以使用布尔条件运算符数据类型，例如 `true` 或 `false`。

例如，`aws:MultiFactorAuthPresent` 全局条件键需要带有 `Boolean` 数据类型的条件运算符。如果您提供日期或整数，则数据类型将不匹配。

相关术语

- 布尔值条件运算符
- IAM JSON 策略元素：条件运算符

常规警告 — 类型不匹配日期

问题代码：TYPE_MISMATCH_DATE

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch date: The date condition operator is used with an invalid value. Specify
 a valid date using YYYY-MM-DD or other ISO 8601 date/time format.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The date condition operator is used with an invalid value. Specify a
 valid date using YYYY-MM-DD or other ISO 8601 date/time format."
```

解决一般警告

请更新文本以使用日期条件运算符数据类型，采用 `YYYY-MM-DD` 或其他 ISO 8601 日期时间格式。

相关术语

- 日期条件运算符

- IAM JSON 策略元素：条件运算符

常规警告 — 类型不匹配的数字

问题代码：TYPE_MISMATCH_NUMBER

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch number: Add a valid numeric value for the condition operator
 {{operator}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid numeric value for the condition operator {{operator}}."
```

解决一般警告

请更新文本以使用数字条件运算符数据类型。

相关术语

- 数字条件运算符
- IAM JSON 策略元素：条件运算符

常规警告 — 类型不匹配的字符串

问题代码：TYPE_MISMATCH_STRING

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Type mismatch string: Add a valid base64-encoded string value for the condition
 operator {{operator}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a valid base64-encoded string value for the condition operator
 {{operator}}."
```

解决一般警告

请更新文本以使用字符串条件运算符数据类型。

相关术语

- 字符串条件运算符
- IAM JSON 策略元素：条件运算符

常规警告 - 建议使用特定的 github 存储库和分支

问题代码：SPECIFIC_GITHUB_REPO_AND_BRANCH_RECOMMENDED

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Specific github repo and branch recommended: Using a wildcard (*) in
 token.actions.githubusercontent.com:sub can allow requests from more sources than
 you intended. Specify the value of token.actions.githubusercontent.com:sub with the
 repository and branch name.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using a wildcard (*) in token.actions.githubusercontent.com:sub
 can allow requests from more sources than you intended. Specify the value of
 token.actions.githubusercontent.com:sub with the repository and branch name."
```

解决一般警告

如果您将 GitHub 用作 OIDC IdP，则最佳实践是限制可以代入与 IAM IdP 关联的角色的实体。如果信任策略中包含 `Condition` 语句，则可将角色限制为特定的 GitHub 组织、存储库或分支。您可以使用

条件键 token.actions.githubusercontent.com:sub 限制访问。我们建议您将条件限制为一组特定的存储库或分支。如果在 token.actions.githubusercontent.com:sub 中使用通配符（*），则来自您控制范围之外的组织或存储库的 GitHub 操作可代入与您 Amazon 账户中的 GitHub IAM IdP 关联的角色。

相关术语

- 为 GitHub OIDC 身份提供商配置角色

常规警告 - 策略大小超出角色信任策略配额

问题代码：POLICY_SIZE_EXCEEDS_ROLE_TRUST_POLICY_QUOTA

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Policy size exceeds role trust policy quota: The characters in the role trust policy,
 excluding whitespace, exceed the character maximum. We recommend that you request a
 role trust policy length quota increase using Service Quotas and AWS Support Center.
 If the quotas have already been increased, then you can ignore this warning.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The characters in the role trust policy, excluding whitespace,
 exceed the character maximum. We recommend that you request a role trust policy length
 quota increase using Service Quotas and AWS Support Center. If the quotas have already
 been increased, then you can ignore this warning."
```

解决一般警告

IAM 与 Amazon STS 具有限制角色信任策略大小的配额。角色信任策略中的字符（不包括空格）超过了最大字符数。我们建议您使用服务限额和 Amazon Support Center Console 请求增加角色信任策略长度配额。

相关术语

- IAM 与 Amazon STS 配额、名称要求和字符限制

一般警告 – RCP 缺少相关的主体条件键

问题代码：RCP_MISSING_RELATED_PRINCIPAL_CONDION_KEY

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP missing related principal condition key: RCPs impact IAM roles, users, and AWS
 service principals. To prevent unintended impact to services acting on your behalf
 using a service principal, an additional statement should be added to the Condition
 block "BoolIfExists": { "aws:PrincipalIsAWSService": "false"} whenever a principal key
 {{conditionKeyName}} is used.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "RCPs impact IAM roles, users, and AWS service principals. To prevent
 unintended impact to services acting on your behalf using a service principal,
 an additional statement should be added to the Condition block "BoolIfExists":
 { "aws:PrincipalIsAWSService": "false"} whenever a principal key {{conditionKeyName}}
 is used."
```

解决一般警告

Amazon Organizations 资源控制策略（RCP）可能会影响 IAM 角色、用户和 Amazon Web Services 服务 主体。为防止使用服务主体代表您行事的服务受到意外影响，请在 `Condition` 元素中添加以下语句：

```
"BoolIfExists": { "aws:PrincipalIsAWSService": "false"}
```

相关术语

- RCP 语法
- 主体的属性

一般警告 – RCP 缺少相关的服务主体条件键

问题代码：RCP_MISSING_RELATED_SERVICE_PRINCIPAL_CONDITION_KEY

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP missing related service principal condition key: RCPs impact IAM roles, users,
 and AWS service principals. To prevent unintended impact to your principals,
 an additional statement should be added to the Condition block "BoolIfExists":
 { "aws:PrincipalIsAWSService": "true"} whenever the key {{conditionKeyName}} is used.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "RCPs impact IAM roles, users, and AWS service principals. To prevent
 unintended impact to your principals, an additional statement should be added to the
 Condition block "BoolIfExists": { "aws:PrincipalIsAWSService": "true"} whenever the
 key {{conditionKeyName}} is used."
```

解决一般警告

Amazon Organizations 资源控制策略（RCP）可能会影响 IAM 角色、用户和 Amazon Web Services 服务 主体。为防止对您的主体造成意外影响，请将以下语句添加到您的 `Condition` 元素：

```
"BoolIfExists": { "aws:PrincipalIsAWSService": "true"}
```

相关术语

- RCP 语法
- 主体的属性

一般警告 – RCP missing service condition key null check

问题代码：RCP_MISSING_SERVICE_CONDION_KEY_NULL_CHECK

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
RCP missing service condition key null check: The specified service may have a service
 integration that does not require the use of the {{conditionKeyName}} condition
 key. To prevent unintended impact to services acting on your behalf using a service
 principal, an additional statement should be added to the Condition block "Null":
 { "aws:SourceAccount": "false"} or "Null": { "aws:SourceArn": "false"} whenever the
 key {{conditionKeyName}} is used.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The specified service may have a service integration that does
 not require the use of the {{conditionKeyName}} condition key. To prevent unintended
 impact to services acting on your behalf using a service principal, an additional
 statement should be added to the Condition block "Null": { "aws:SourceAccount":
 "false"} or "Null": { "aws:SourceArn": "false"} whenever the key {{conditionKeyName}}
 is used."
```

解决一般警告

Amazon Organizations 资源控制策略（RCP）可能会影响 IAM 角色、用户和 Amazon Web Services 服务 主体。为防止使用服务主体代表您行事的服务受到意外影响，请在使用指定键时将以下语句之一添加到您的 Condition 元素中：

```
"Null": { "aws:SourceAccount": "false"}
```

或者

```
"Null": { "aws:SourceArn": "false"}
```

相关术语

- RCP 语法
- 主体的属性

一般警告 - 条件键仅与支持的服务搭配使用

问题代码：USE_CONDITION_KEY_ONLY_WITH_SUPPORTED_SERVICES

调查发现类型：GENERAL_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Use condition key only with supported services: The condition key {{key}} works only
 with specific AWS services and must be scoped to supported services in your policies.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{key}} works only with specific AWS services and
 must be scoped to supported services in your policies.
```

解决一般警告

查看 Amazon 文档以确定哪些 Amazon Web Services 服务 支持此条件键。如果策略中的任何服务不支持条件键，请修改策略，将条件键的范围限定为仅支持条件键的 Amazon Web Services 服务 服务。

相关术语

- aws:VpceAccount
- aws:VpceOrgID
- aws:VpceOrgPaths
- Amazon Web Services 服务 的操作、资源和条件键

安全警告 – 不可信的条件键

问题代码：UNTRUSTWORTHY_CONDITION_KEY

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Untrustworthy condition key: The {{key}} condition key is not recommended for access
 control as it can be spoofed/manipulated by the caller.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{key}} condition key is not recommended for access control as
 it can be spoofed/manipulated by the caller."
```

解决安全警告

请勿将此条件键用于访问控制。调用者有可能会篡改或伪造密钥值，这会带来安全隐患。

相关术语

- Amazon 全局条件上下文密钥

安全警告 — 允许使用 NotPrincipal

问题代码：ALLOW_WITH_NOT_PRINCIPAL

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Allow with NotPrincipal: Using Allow with NotPrincipal can be overly permissive. We
 recommend that you use Principal instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using Allow with NotPrincipal can be overly permissive. We recommend
 that you use Principal instead."
```

解决安全警告

使用 `"Effect": "Allow"` 与 `NotPrincipal` 可能会过于宽松。例如，这可以向匿名主体授予权限。Amazon 会建议您使用 `Principal` 元素指定需要访问权限的主体。或者，您可以允许广泛访问，然后添加另一个使用带有 `“Effect”: “Deny”` 的 `NotPrincipal` 元素的语句。

- Amazon JSON 策略元素：Principal
- Amazon JSON 策略元素：NotPrincipal

安全警告 — 带有单值密钥的 ForAllValues

问题代码：FORALLVALUES_WITH_SINGLE_VALUED_KEY

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
ForAllValues with single valued key: Using ForAllValues qualifier with the single-
valued condition key {{key}} can be overly permissive. We recommend that you remove
 ForAllValues:.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using ForAllValues qualifier with the single-valued condition key
 {{key}} can be overly permissive. We recommend that you remove ForAllValues:."
```

解决安全警告

Amazon 会建议您仅将 `ForAllValues` 用于多值条件。`ForAllValues` 集合运算符测试请求集的每个成员的值是否为条件键集的子集。如果请求中的每个键值均与策略中的至少一个值匹配，则条件返回 true。如果请求中没有键或者键值解析为空数据集（如空字符串），则也会返回 true。

要了解条件是支持单个值还是多个值，请查看该服务的操作、资源和条件密钥页面。带有 `ArrayOf` 数据类型前缀的条件密钥是多值条件密钥。例如，Amazon SES 支持具有单个值 (`String`) 的密钥和 `ArrayOfString` 多值数据类型。

- 多值上下文键

安全警告 — 使用 NotResource 传递角色

问题代码：PASS_ROLE_WITH_NOT_RESOURCE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with NotResource: Using the iam:PassRole action with NotResource can be
 overly permissive because it can allow iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:PassRole action with NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在带有 `NotResource` 元素的策略中使用 `iam:PassRole` 可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- 将角色传递给服务
- iam:PassedToService
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Resource

安全警告 — 使用带有星号的操作和 NotResource 传递角色

问题代码：PASS_ROLE_WITH_STAR_IN_ACTION_AND_NOT_RESOURCE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with star in action and NotResource: Using an action with a wildcard (*) and
 NotResource can be overly permissive because it can allow iam:PassRole permissions on
 multiple resources. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using an action with a wildcard (*) and NotResource can be overly
 permissive because it can allow iam:PassRole permissions on multiple resources. We
 recommend that you specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在 `Action` 中带有通配符 (*) 且包含 `NotResource` 元素的策略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- 将角色传递给服务
- iam:PassedToService
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Resource

安全警告 — 使用 NotAction 和 NotResource 传递角色

问题代码：PASS_ROLE_WITH_NOT_ACTION_AND_NOT_RESOURCE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with NotAction and NotResource: Using NotAction with NotResource can
 be overly permissive because it can allow iam:PassRole permissions on multiple
 resources.. We recommend that you specify resource ARNs instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using NotAction with NotResource can be overly permissive because
 it can allow iam:PassRole permissions on multiple resources.. We recommend that you
 specify resource ARNs instead."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。使用 `NotAction` 元素并列出 `NotResource` 元素中的一些资源可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- 将角色传递给服务
- iam:PassedToService
- IAM JSON 策略元素：NotAction
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Resource

安全警告 — 使用带有星号的资源传递角色

问题代码：PASS_ROLE_WITH_STAR_IN_RESOURCE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with star in resource: Using the iam:PassRole action with wildcards (*)
 in the resource can be overly permissive because it allows iam:PassRole permissions
 on multiple resources. We recommend that you specify resource ARNs or add the
 iam:PassedToService condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the iam:PassRole action with wildcards (*) in the resource can
 be overly permissive because it allows iam:PassRole permissions on multiple resources.
 We recommend that you specify resource ARNs or add the iam:PassedToService condition
 key to your statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。允许 `iam:PassRole` 且在 `Resource` 元素中包含通配符 (*) 的策略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

某些 Amazon 服务在其角色名称中包含服务命名空间。此策略检查会在分析策略以生成结果时考虑到这些约定。例如，以下资源 ARN 可能不会生成结果：

```
arn:aws:iam::*:role/Service*
```

- 将角色传递给服务
- iam:PassedToService
- IAM JSON 策略元素：Resource

带有此安全警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一个使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限，并授予将任何 IAM 角色传递给任何服务的权限。Amazon 会建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- AdministratorAccess-Amplify

以下 Amazon 托管策略包括对 `iam:PassRole` 的权限，并在资源包含通配符 (*)，且位于弃用路径。对于这每一个策略，我们都更新了权限指南，例如推荐一个新的 Amazon 托管策略，以支持使用案例。要查看这些策略的替代方案，请参阅每项服务的指南。

- AWSElasticBeanstalkFullAccess
- AWSElasticBeanstalkService
- AWSLambdaFullAccess
- AWSLambdaReadOnlyAccess
- AWSOpsWorksFullAccess
- AWSOpsWorksRole
- AWSDataPipelineRole
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonElasticMapReduceFullAccess
- AmazonDynamoDBFullAccesswithDataPipeline
- AmazonEC2ContainerServiceFullAccess

以下 Amazon 托管策略仅向服务关联角色提供权限，它允许 Amazon 服务代表您执行操作。您可以将策略附加得到 IAM 身份。

- AWSServiceRoleForAmazonEKSNodegroup

安全警告 — 使用带有星号的行动和资源传递角色

问题代码：PASS_ROLE_WITH_STAR_IN_ACTION_AND_RESOURCE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with star in action and resource: Using wildcards (*) in the action and the
 resource can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using wildcards (*) in the action and the resource can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在 `Action` 中带有通配符 (*) 且包含 `Resource` 元素的策略可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- 将角色传递给服务
- iam:PassedToService
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：Resource

带有此安全警告的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

其中一些使用案例适用于您账户中的管理员。以下 Amazon 托管策略提供管理员访问权限，并授予将任何 IAM 角色传递给任何 Amazon 服务的权限。Amazon 会建议您仅将以下 Amazon 托管策略附加到您认为是管理员的 IAM 身份。

- AdministratorAccess
- IAMFullAccess

安全警告 — 使用带有星号的资源和 NotAction 传递角色

问题代码：PASS_ROLE_WITH_STAR_IN_RESOURCE_AND_NOT_ACTION

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Pass role with star in resource and NotAction: Using a resource with wildcards (*) and
 NotAction can be overly permissive because it allows iam:PassRole permissions on all
 resources. We recommend that you specify resource ARNs or add the iam:PassedToService
 condition key to your statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using a resource with wildcards (*) and NotAction can be overly
 permissive because it allows iam:PassRole permissions on all resources. We recommend
 that you specify resource ARNs or add the iam:PassedToService condition key to your
 statement."
```

解决安全警告

要配置多项 Amazon 服务，您必须将 IAM 角色传递 给相应服务。要允许这样做，您必须对身份（用户、用户组或角色）授予 `iam:PassRole` 的权限。在策略中使用 `NotAction` 元素且在 `Resource` 元素中包含通配符 (*)可以允许您的主体访问比您预期更多的服务或功能。Amazon 会建议您在 `Resource` 元素中指定允许的 ARN。此外，您可以通过使用 `iam:PassedToService` 条件键减少对单个服务的权限。

- 将角色传递给服务
- iam:PassedToService

- IAM JSON 策略元素：NotAction
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：Resource

安全警告 — 缺少配对条件键

问题代码：MISSING_PAIRED_CONDITION_KEYS

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing paired condition keys: Using the condition key {{conditionKeyName}}
 can be overly permissive without also using the following condition keys:
 {{recommendedKeys}}. Condition keys like this one are more secure when paired with
 a related key. We recommend that you add the related condition keys to the same
 condition block.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the condition key {{conditionKeyName}} can be overly
 permissive without also using the following condition keys: {{recommendedKeys}}.
 Condition keys like this one are more secure when paired with a related key. We
 recommend that you add the related condition keys to the same condition block."
```

解决安全警告

当与其他相关条件键配对时，某些条件键会更加安全。Amazon 建议您将相关条件键包含在与现有条件键相同的条件数据块中。这会使得通过策略授予的权限更加安全。

例如，您可以使用 `aws:VpcSourceIp` 条件键将发出请求的 IP 地址与您在策略中指定的 IP 地址进行比较。Amazon 会建议您添加相关 `aws:SourceVPC` 条件键。这将检查请求是否来自您在策略中指定的 VPC 以及您指定的 IP 地址。

相关术语

- `aws:VpcSourceIp` 全局条件键

- aws:SourceVPC 全局条件键
- 全局条件键
- 条件元素
- JSON 策略概述

安全警告 — 使用不支持的标签条件键拒绝服务

问题代码：DENY_WITH_UNSUPPORTED_TAG_CONDION_KEY_FOR SERVICE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Deny with unsupported tag condition key for service: Using the effect Deny with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes can be overly permissive: {{serviceNames}}. Actions for the listed services
 are not denied by this statement. We recommend that you move these actions to a
 different statement without this condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Deny with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes can be
 overly permissive: {{serviceNames}}. Actions for the listed services are not denied
 by this statement. We recommend that you move these actions to a different statement
 without this condition key."
```

解决安全警告

在带有 "Effect": "Deny" 的政策的 Condition 元素中使用不受支持的标签条件键可能会过于宽松，因为该服务的条件将被忽略。Amazon 会建议您删除不支持条件键的服务操作，并创建另一个语句来拒绝对这些操作的特定资源的访问。

如果您使用 aws:ResourceTag 条件键，并且服务操作不支持该键，则该键不会包含在请求上下文中。在本例中，Deny 语句中的条件总是返回 false，并且该行动从来没有被拒绝。即使资源被正确标记，也会发生这种情况。

当服务支持 aws:ResourceTag 条件键时，您可以使用标签来控制对该服务资源的访问。这称为基于属性的访问控制 (ABAC)。不支持这些密钥的服务需要您使用基于资源的访问控制 (RBAC) 来控制对资源的访问。

> **Note**
>
> 某些服务允许支持其资源和操作子集的 aws:ResourceTag 条件键。IAM Access Analyzer 将返回不受支持的服务操作结果。例如，Amazon S3 支持其资源子集的 aws:ResourceTag。要查看 Amazon S3 中支持 aws:ResourceTag 条件键的所有资源类型，请参阅服务授权参考中的 Amazon S3 定义的资源类型。

例如，假设您想拒绝访问，以将被删除但使用了键值对 status=Confidential 进行标记的特定资源解除标记。同时假设 Amazon Lambda 允许您标记和取消标记资源，但不支持 aws:ResourceTag 条件键。要在此标签存在的情况下拒绝 Amazon App Mesh 和 Amazon Backup 的删除操作，请使用 aws:ResourceTag 条件键。对于 Lambda，请使用包含 "Confidential" 前缀的资源命名约定。然后包含一个单独的语句，以防止使用该命名约定删除资源。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenyDeleteSupported",
            "Effect": "Deny",
            "Action": [
                "appmesh:DeleteMesh",
                "backup:DeleteBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/status": "Confidential"
                }
            }
        },
        {
            "Sid": "DenyDeleteUnsupported",
            "Effect": "Deny",
            "Action": "lambda:DeleteFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:status-Confidential*"
        }
```

```
    ]
}
```

> **Warning**
>
> 请勿将...IfExists版本的条件运算符用作此结果的解决方法。这表示“当键存在于请求上下文中且值匹配时，拒绝操作。否则，拒绝操作。” 在前面的示例中，在带有 `StringEqualsIfExists` 运算符的 `DenyDeleteSupported` 语句中包含 `lambda:DeleteFunction` 操作将始终拒绝该操作。对于该操作，密钥不存在于上下文中，并且每次尝试删除该资源类型时都会被拒绝，无论资源是否被标记。

相关术语

- 全局条件键
- 比较 ABAC 与 RBAC
- IAM JSON 策略元素：条件运算符
- 条件元素
- JSON 策略概述

安全警告 — 使用不支持的标签条件键拒绝服务的 NotAction

问题代码：DENY_NOTACTION_WITH_UNSUPPORTED_TAG_CONDITION_KEY_FOR_SERVICE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Deny NotAction with unsupported tag condition key for service: Using the effect Deny
 with NotAction and the tag condition key {{conditionKeyName}} can be overly permissive
 because some service actions are not denied by this statement. This is because the
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Deny with NotAction and the tag condition key
 {{conditionKeyName}} can be overly permissive because some service actions are not
 denied by this statement. This is because the condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction."
```

解决安全警告

在带有元素 NotAction 和 "Effect": "Deny" 的策略的 Condition 元素中使用标签条件键可能会过于宽松。对于不支持条件键的服务操作，该条件将被忽略。Amazon 会建议您重写逻辑以拒绝操作列表。

如果您将 aws:ResourceTag 条件键与 NotAction 搭配使用，则不会拒绝任何不支持密钥的新或现有服务操作。Amazon 会建议您明确列出要拒绝的操作。IAM Access Analyzer 为列出的不支持 aws:ResourceTag 条件键的操作返回单独的结果。有关更多信息，请参阅 安全警告 — 使用不支持的标签条件键拒绝服务。

当服务支持 aws:ResourceTag 条件键时，您可以使用标签来控制对该服务资源的访问。这称为基于属性的访问控制 (ABAC)。不支持这些密钥的服务需要您使用基于资源的访问控制 (RBAC) 来控制对资源的访问。

相关术语

- 全局条件键
- 比较 ABAC 与 RBAC
- IAM JSON 策略元素：条件运算符
- 条件元素
- JSON 策略概述

安全警告 — 限制对服务主体的访问

问题代码：RESTRICT_ACCESS_TO_SERVICE_PRINCIPAL

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Restrict access to service principal: Granting access to a service principal without
 specifying a source is overly permissive. Use aws:SourceArn, aws:SourceAccount,
 aws:SourceOrgID, or aws:SourceOrgPaths condition key to grant fine-grained access.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Granting access to a service principal without specifying a source
 is overly permissive. Use aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or
 aws:SourceOrgPaths condition key to grant fine-grained access."
```

解决安全警告

您可以使用服务主体指定基于资源策略的 `Principal` 元素中的 Amazon Web Services 服务，即服务的标识符。授予服务主体的访问权限以代表您行事时，请限制访问权限。您可以通过 `aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID` 或 `aws:SourceOrgPaths` 条件键来限制特定资源 ARN、Amazon Web Services 账户、组织 ID 或组织路径等特定来源的访问，从而避免过于宽松的策略。限制访问可以帮助您防止称为混淆代理问题的安全问题。

相关术语

- Amazon Web Services 服务 主体
- Amazon 全局条件密钥：aws:SourceAccount
- Amazon 全局条件密钥：aws:SourceArn
- Amazon 全局条件键：aws:SourceOrgId
- Amazon 全局条件键：aws:SourceOrgPaths
- 混淆代理人问题

安全警告 - 缺少 oidc 主体的条件键

问题代码：MISSING_CONDION_KEY_FOR_OIDC_PRINCIPAL

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing condition key for oidc principal: Using an Open ID Connect principal without a
 condition can be overly permissive. Add condition keys with a prefix that matches your
```

```
federated OIDC principals to ensure that only the intended identity provider assumes
the role.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using an Open ID Connect principal without a condition can be
overly permissive. Add condition keys with a prefix that matches your federated OIDC
principals to ensure that only the intended identity provider assumes the role."
```

解决安全警告

无条件地使用 Open ID Connect 主体可能过于宽容。添加前缀与联合 OIDC 主体匹配的条件键，以确保只有预期的身份提供商能够代入角色。

相关术语

- 创建用于 Web 身份或 OpenID Connect 联合身份验证的角色（控制台）

安全警告 - 缺少 github 存储库条件键

问题代码：MISSING_GITHUB_REPO_CONDITION_KEY

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Missing github repo condition key: Granting a federated GitHub principal permissions
without a condition key can allow more sources to assume the role than you intended.
Add the token.actions.githubusercontent.com:sub condition key and specify the branch
and repository name in the value.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Granting a federated GitHub principal permissions without a
condition key can allow more sources to assume the role than you intended. Add the
token.actions.githubusercontent.com:sub condition key and specify the branch and
repository name in the value."
```

解决安全警告

如果您将 GitHub 用作 OIDC IdP，则最佳实践是限制可以代入与 IAM IdP 关联的角色的实体。如果信任策略中包含 Condition 语句，则可将角色限制为特定的 GitHub 组织、存储库或分支。您可以使用条件键 token.actions.githubusercontent.com:sub 限制访问。我们建议您将条件限制为一组特定的存储库或分支。如果您未包含此条件，则来自您控制范围之外的组织或存储库的 GitHub 操作可代入与您 Amazon 账户中的 GitHub IAM IdP 关联的角色。

相关术语

- 为 GitHub OIDC 身份提供商配置角色

安全警告 – String like operator with ARN condition keys

问题代码：STRING_LIKE_OPERATOR_WITH_ARN_CONDITION_KEYS

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
String like operator with ARN condition keys: Use the operator type {{allowed}} instead
 of operator {{operator}} for the condition key {{key}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Use the operator type {{allowed}} instead of operator {{operator}}
 for the condition key {{key}}."
```

解决安全警告

Amazon 建议您在比较 ARN 时使用 ARN 运算符而不是字符串运算符，以确保根据 ARN 条件值进行适当的访问限制。每当使用指定的键时，请将 Condition 元素中的 StringLike 运算符更新为 ArnLike 运算符。

以下 Amazon 托管式策略是此安全警告的例外情况：

- AmazonSecurityLakeAdministrator
- AWSCodePipeline_FullAccess

- AWSCodePipeline_ReadOnlyAccess
- S3UnlockBucketPolicy
- SQSUnlockQueuePolicy

相关术语

- Amazon 资源名称（ARN）条件运算符
- 字符串条件运算符
- Amazon 托管式策略

安全警告 — ForAnyValue（含受众声明类型）

问题代码：FFORANYVALUE_WITH_AUDIENCE_CLAIM_TYPE

调查发现类型：SECURITY_WARNING

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
ForAnyValue with audience claim type: Using ForAnyValue qualifier with the single-
valued condition key {{key}} can be overly permissive. We recommend that you remove
 ForAnyValue:.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using ForAnyValue qualifier with the single-valued condition key
 {{key}} can be overly permissive. We recommend that you remove ForAnyValue:."
```

解决安全警告

Amazon 建议不要将条件集合运算符 `ForAnyValue` 用于单值条件键。仅将集合运算符用于多值条件键。移除 `ForAnyValue` 集合运算符。

相关术语

- 单值和多值上下文键
- 单值上下文键策略示例

建议 — 空数组操作

问题代码：EMPTY_ARRAY_ACTION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array action: This statement includes no actions and does not affect the policy.
 Specify actions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no actions and does not affect the policy.
 Specify actions."
```

解决建议

语句必须包含 `Action` 或者 `NotAction` 元素，其中包含一组操作。当元素为空时，策略语句不提供任何权限。在 `Action` 元素中指定操作。

- IAM JSON 策略元素：Action

建议 — 空数组条件

问题代码：EMPTY_ARRAY_CONDITION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array condition: There are no values for the condition key {{key}} and it does
 not affect the policy. Specify conditions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "There are no values for the condition key {{key}} and it does not
 affect the policy. Specify conditions."
```

解决建议

可选 Condition 元素结构要求您使用条件运算符和键值对。当条件值为空时，条件返回 true 并且策略语句不再提供任何权限。指定条件值。

- IAM JSON 策略元素：Condition

建议 — 空数组条件 ForAllValues

问题代码：EMPTY_ARRAY_CONDITION_FORALLVALUES

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array condition ForAllValues: The ForAllValues prefix with an empty condition key
 matches only if the key {{key}} is missing from the request context. To determine if
 the request context is empty, we recommend that you use the Null condition operator
 with the value of true instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The ForAllValues prefix with an empty condition key matches only
 if the key {{key}} is missing from the request context. To determine if the request
 context is empty, we recommend that you use the Null condition operator with the value
 of true instead."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。ForAllValues 集合运算符测试请求集的每个成员的值是否为条件键集的子集。

当您将 ForAllValues 用于空数组条件键时，则只有在请求中没有密钥时，条件才匹配。Amazon 会建议如果要测试请求上下文是否为空，请使用 Null 条件运算符。

- 多值上下文键

- Null 条件运算符
- IAM JSON 策略元素：Condition

建议 — 空数组条件 ForAnyValue

问题代码：EMPTY_ARRAY_CONDITION_FORANYVALUE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array condition ForAnyValue: The ForAnyValue prefix with an empty condition key
 {{key}} never matches the request context and it does not affect the policy. Specify
 conditions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The ForAnyValue prefix with an empty condition key {{key}} never
 matches the request context and it does not affect the policy. Specify conditions."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。ForAnyValues 集合运算符测试请求值集的至少一个成员是否与条件键值集的至少一个成员匹配。

当您使用带有空条件键的 ForAnyValues 时，条件永远不会匹配。这表示语句对策略没有影响。Amazon 会建议您重写条件。

- 多值上下文键
- IAM JSON 策略元素：Condition

建议 — 空数组条件 IfExists

问题代码：EMPTY_ARRAY_CONDITION_IFEXISTS

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array condition IfExists: The IfExists suffix with an empty condition key matches
 only if the key {{key}} is missing from the request context. To determine if the
 request context is empty, we recommend that you use the Null condition operator with
 the value of true instead.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The IfExists suffix with an empty condition key matches only if the
 key {{key}} is missing from the request context. To determine if the request context
 is empty, we recommend that you use the Null condition operator with the value of true
 instead."
```

解决建议

`...IfExists` 后缀编辑条件运算符。这意味着如果请求的内容中存在策略键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 true。

当您将 `...IfExists` 用于空数组条件键时，则只有在请求中没有密钥时，条件才匹配。Amazon 会建议如果要测试请求上下文是否为空，请使用 `Null` 条件运算符。

- ...IfExists 条件运算符
- IAM JSON 策略元素：Condition

建议 — 空数组主体

问题代码：EMPTY_ARRAY_PRINCIPAL

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array principal: This statement includes no principals and does not affect the
 policy. Specify principals.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no principals and does not affect the
 policy. Specify principals."
```

解决建议

您必在 IAM 角色的信任策略和基于资源的策略中使用 `Principal` 或 `NotPrincipal` 元素。基于资源的策略是直接嵌入资源中的策略。

当您在语句的 `Principal` 元素中提供空数组时，语句对策略没有影响。Amazon 会建议您指定应有权访问资源的主体。

- IAM JSON 策略元素：Principal
- IAM JSON 策略元素：NotPrincipal

建议 — 空数组资源

问题代码：EMPTY_ARRAY_RESOURCE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty array resource: This statement includes no resources and does not affect the
 policy. Specify resources.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no resources and does not affect the policy.
 Specify resources."
```

解决建议

语句必须包含 `Resource` 或 `NotResource` 元素。

当您在语句的资源元素中提供空数组时，该语句对策略没有影响。Amazon 会建议您为资源指定 Amazon Resource Names (ARN)。

- IAM JSON 策略元素：Resource

- IAM JSON 策略元素：NotResource

建议 — 空对象条件

问题代码：EMPTY_OBJECT_CONDITION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty object condition: This condition block is empty and it does not affect the
 policy. Specify conditions.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This condition block is empty and it does not affect the policy.
 Specify conditions."
```

解决建议

Condition 元素结构要求您使用条件运算符和键值对。

当您在语句的条件元素中提供空对象时，该语句对策略没有影响。请删除可选元素或指定条件。

- IAM JSON 策略元素：Condition

建议 — 空对象主体

问题代码：EMPTY_OBJECT_PRINCIPAL

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty object principal: This statement includes no principals and does not affect the
 policy. Specify principals.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "This statement includes no principals and does not affect the
 policy. Specify principals."
```

解决建议

您必在 IAM 角色的信任策略和基于资源的策略中使用 `Principal` 或 `NotPrincipal` 元素。基于资源的策略是直接嵌入资源中的策略。

当您在语句的 `Principal` 元素中提供空对象时，语句对策略没有影响。Amazon 会建议您指定应有权访问资源的主体。

- IAM JSON 策略元素：Principal
- IAM JSON 策略元素：NotPrincipal

建议 — 空 Sid 值

问题代码：EMPTY_SID_VALUE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Empty Sid value: Add a value to the empty string in the Sid element.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Add a value to the empty string in the Sid element."
```

解决建议

`Sid`（声明 ID）元素允许您针对策略语句输入标识符。您可以为语句数组中的每个语句指定 `Sid` 值。如果您选择使用 `Sid` 元素，则您必须提供字符串值。

相关术语

- IAM JSON 策略元素：Sid

建议 – 相当于 null 值 false

问题代码：EQUIVALENT_TO_NULL_FALSE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Equivalent to null false: We recommend replacing the key {{key}} in the condition block
 of {{operator}} with {{{recommendedKey}}: false} to ensure better enforcement of the
 condition.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend replacing the key {{key}} in the condition block of
 {{operator}} with {{{recommendedKey}}: false} to ensure better enforcement of the
 condition."
```

解决建议

将当前条件键替换为设置成 `false` 的推荐键。这一更改提高了策略的清晰度，并确保了条件评估更加可靠。更新您的条件块以使用 `{recommendedKey}: false`，而不是当前的键运算符组合。

相关术语

- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥

建议 – 相当于 null 值 true

问题代码：EQUIVALENT_TO_NULL_TRUE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Equivalent to null true: We recommend replacing the key {{key}} in the condition block
 of {{operator}} with {{{recommendedKey}}: true} to ensure better enforcement of the
 condition.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "We recommend replacing the key {{key}} in the condition block of
 {{operator}} with {{{recommendedKey}}: true} to ensure better enforcement of the
 condition."
```

解决建议

将当前条件键替换为设置成 `true` 的推荐键。这一更改提高了策略的清晰度，并确保了条件评估更加可靠。更新您的条件块以使用 `{recommendedKey}: true`，而不是当前的键运算符组合。

相关术语

- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥

建议 — 改善 IP 范围

问题代码：IMPROVE_IP_RANGE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Improve IP range: The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The non-zero bits in the IP address after the masked bits are
 ignored. Replace address with {{addr}}."
```

解决建议

IP 地址条件必须采用标准的 CIDR 格式，例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64。如果在屏蔽位之后包含非零位，则不考虑这些位作为条件。Amazon 会建议您使用消息中包含的新地址。

- IP 地址条件运算符
- IAM JSON 策略元素：Condition

建议 — 带限定符的空值

问题代码：NULL_WITH_QUALIFIER

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Null with qualifier: Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Avoid using the Null condition operator with the ForAllValues or
 ForAnyValue qualifiers because they always return a true or false respectively."
```

解决建议

在 Condition 元素中，您可构建表达式并使用条件运算符（等于、小于等）将策略中的条件键和值与请求上下文中的键和值进行匹配。对单个条件键包含多个值的请求而言，必须使用 ForAllValues 或 ForAnyValue 集合运算符。

当您使用带有 ForAllValues 的 Null 条件运算符时，则该语句将始终返回 true。当您使用带有 ForAnyValue 的 Null 条件运算符时，则该语句始终返回 false。Amazon 会建议您将 StringLike 条件运算符与这些集合运算符搭配使用。

相关术语

- 多值上下文键

- Null 条件运算符
- 条件元素

建议 — 私有 IP 地址子集

问题代码：PRIVATE_IP_ADDRESS_SUBSET

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Private IP address subset: The values for condition key aws:SourceIp include a mix
 of private and public IP addresses. The private addresses will not have the desired
 effect. aws:SourceIp works only for public IP address ranges. To define permissions
 for private IP ranges, use aws:VpcSourceIp.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses will not have the desired effect.
 aws:SourceIp works only for public IP address ranges. To define permissions for
 private IP ranges, use aws:VpcSourceIp."
```

解决建议

全局条件键 `aws:SourceIp` 仅适用于公有 IP 地址范围。

当您的 `Condition` 元素同时包含私有 IP 地址和公有 IP 地址时，语句可能不会产生所需的效果。您不能使用 `aws:VpcSourceIP` 指定私有 IP 地址。

> **Note**
>
> 全局条件键 `aws:VpcSourceIP` 仅在请求来自指定的 IP 地址并经过 VPC 终端节点时进行匹配。

- aws:SourceIp 全局条件键

- aws:VpcSourceIp 全局条件键
- IP 地址条件运算符
- IAM JSON 策略元素：Condition

建议 — 私有 NotIpAddress 子集

问题代码：PRIVATE_NOT_IP_ADDRESS_SUBSET

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Private NotIpAddress subset: The values for condition key aws:SourceIp include a mix of
 private and public IP addresses. The private addresses have no effect. aws:SourceIp
 works only for public IP address ranges. To define permissions for private IP ranges,
 use aws:VpcSourceIp.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The values for condition key aws:SourceIp include a mix of private
 and public IP addresses. The private addresses have no effect. aws:SourceIp works
 only for public IP address ranges. To define permissions for private IP ranges, use
 aws:VpcSourceIp."
```

解决建议

全局条件键 `aws:SourceIp` 仅适用于公有 IP 地址范围。

当您的 `Condition` 元素包含 `NotIpAddress` 条件运算符以及同时包括私有 IP 地址和公有 IP 地址的时，语句可能不会产生所需的效果。策略中未指定的每个公有 IP 地址都将匹配。任何私有 IP 地址都不会匹配。为了达到这个效果，您可以使用带有 `aws:VpcSourceIP` 的 `NotIpAddress` 并指定不应匹配的私有 IP 地址。

- aws:SourceIp 全局条件键
- aws:VpcSourceIp 全局条件键
- IP 地址条件运算符
- IAM JSON 策略元素：Condition

建议 — 冗余操作

问题代码：REDUNDANT_ACTION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant action: The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{redundantActionCount}} action(s) are redundant because they
 provide similar permissions. Update the policy to remove the redundant action such as:
 {{redundantAction}}."
```

解决建议

当在 `Action` 元素中使用通配符 (*) 时，您可以包含冗余权限。Amazon 会建议您查看策略并仅包含所需权限。这可以帮助您删除冗余操作。

例如，以下操作包括两次 `iam:GetCredentialReport` 操作。

```
"Action": [
        "iam:Get*",
        "iam:List*",
        "iam:GetCredentialReport"
    ],
```

在此示例中，为以 `Get` 或 `List` 开头的每个 IAM 操作均定义了权限。当 IAM 添加额外的获取或列示操作时，此策略将允许这些操作。您可能希望允许所有这些只读操作。`iam:GetCredentialReport` 操作已作为 `iam:Get*` 的一部分纳入。要删除重复的权限，您可以删除 `iam:GetCredentialReport`。

当操作的所有内容都是冗余时，您会收到此策略检查的结果。在此示例中，如果元素包含 `iam:*CredentialReport`，则不被认为是冗余的。这包括冗余的 `iam:GetCredentialReport` 和

非冗余的 iam:GenerateCredentialReport。删除 iam:Get* 或 iam:*CredentialReport 将更改策略的权限。

- IAM JSON 策略元素：Action

带有此建议的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

冗余操作不影响策略所授予的权限。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您从策略中删除冗余操作。

建议 — 冗余条件值编号

问题代码：REDUNDANT_CONDION_VALUE_NUM

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant condition value num: Multiple values in {{operator}} are redundant. Replace
 with the {{greatest/least}} single value for {{key}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Multiple values in {{operator}} are redundant. Replace with the
 {{greatest/least}} single value for {{key}}."
```

解决建议

对条件键中的类似值使用数字条件运算符时，您可以创建重叠，从而会导致冗余权限。

例如，以下 Condition 元素包含多个具有 1200 秒年龄重叠的 aws:MultiFactorAuthAge 条件。

```
"Condition": {
        "NumericLessThan": {
          "aws:MultiFactorAuthAge": [
            "2700",
            "3600"
          ]
```

```
        }
      }
```

在此示例中，如果在不到 3600 秒（1 小时）之前完成多重身份验证 (MFA)，则定义权限。您可以删除冗余 2700 值。

- 数字条件运算符
- IAM JSON 策略元素：Condition

建议 — 冗余资源

问题代码：REDUNDANT_RESOURCE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant resource: The {{redundantResourceCount}} resource ARN(s) are redundant
 because they reference the same resource. Review the use of wildcards (*)
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The {{redundantResourceCount}} resource ARN(s) are redundant because
 they reference the same resource. Review the use of wildcards (*)"
```

解决建议

当您在 Amazon Resource Names (ARN) 中使用通配符 (*) 时，您可以创建冗余资源权限。

例如，以下 Resource 元素包含多个具有冗余权限的 ARN。

```
"Resource": [
            "arn:aws:iam::111122223333:role/jane-admin",
            "arn:aws:iam::111122223333:role/jane-s3only",
            "arn:aws:iam::111122223333:role/jane*"
        ],
```

在此示例中，为名称以 jane 开头的任何角色定义了权限。您可以删除冗余 jane-admin 和 jane-s3only ARN，而不更改生成的权限。这确实会使策略变得动态化。它将为未来的任何以 jane 开头的

角色定义权限。如果策略的目的是允许访问静态数量的角色，则删除最后一个 ARN 并仅列出应定义的 ARN。

- IAM JSON 策略元素：Resource

带有此建议的 Amazon 托管策略

Amazon 托管策略将根据一般 Amazon 使用案例分配权限，从而使您能开始使用 Amazon。

冗余资源不影响策略所授予的权限。使用 Amazon 托管策略作为创建客户托管策略的参考时，Amazon 会建议您从策略中删除冗余资源。

建议 — 冗余语句

问题代码：REDUNDANT_STATEMENT

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant statement: The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The statements are redundant because they provide identical
 permissions. Update the policy to remove the redundant statement."
```

解决建议

`Statement` 元素为策略的主要元素。该元素为必填项。`Statement` 元素可以包含一条语句或由单独语句组成的数组。

如果在长策略中多次包含相同的语句，则为冗余语句。您可以删除其中一个语句，不会影响策略授予的权限。当某人编辑策略时，他们可能会更改其中一个语句而不更新副本。这可能会导致超出预期的权限。

- IAM JSON 策略元素：Statement

建议 — 服务名称中的通配符

问题代码：WILDCARD_IN_SERVICE_NAME

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Wildcard in service name: Avoid using wildcards (*, ?) in the service name because it
 might grant unintended access to other AWS services with similar names.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Avoid using wildcards (*, ?) in the service name because it might
 grant unintended access to other AWS services with similar names."
```

解决建议

当您在策略中包含 Amazon 服务的名称时，Amazon 建议您不要包括通配符 (*, ?)。这可能会为您意料之外的未来服务添加权限。例如，有数十个 Amazon 服务在其名称中包含 `*code*`。

```
"Resource": "arn:aws:*code*::111122223333:*"
```

- IAM JSON 策略元素：Resource

建议 — 使用不支持的标签条件键允许服务

问题代码：ALLOW_WITH_UNSUPPORTED_TAG_CONDITION_KEY_FOR_SERVICE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Allow with unsupported tag condition key for service: Using the effect Allow with the
 tag condition key {{conditionKeyName}} and actions for services with the following
 prefixes does not affect the policy: {{serviceNames}}. Actions for the listed service
```

```
are not allowed by this statement. We recommend that you move these actions to a
different statement without this condition key.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Allow with the tag condition key
 {{conditionKeyName}} and actions for services with the following prefixes does not
 affect the policy: {{serviceNames}}. Actions for the listed service are not allowed
 by this statement. We recommend that you move these actions to a different statement
 without this condition key."
```

解决建议

在带有 `"Effect": "Allow"` 的策略的 `Condition` 元素中使用不受支持的标签条件键不会影响策略授予的权限，因为该服务操作将忽略该条件。Amazon 会建议您删除将不支持条件键的服务的操作删除，并创建另一个语句以允许访问该服务中的特定资源。

如果您使用 `aws:ResourceTag` 条件键，并且服务操作不支持该键，则该键不会包含在请求上下文中。在本例中，`Allow` 语句中的条件总是返回 `false` 并且该操作将永远不被允许。即使资源被正确标记，也会发生这种情况。

当服务支持 `aws:ResourceTag` 条件键时，您可以使用标签来控制对该服务资源的访问。这称为基于属性的访问控制 (ABAC)。不支持这些密钥的服务需要您使用基于资源的访问控制 (RBAC) 来控制对资源的访问。

> **Note**
>
> 某些服务允许支持其资源和操作子集的 `aws:ResourceTag` 条件键。IAM Access Analyzer 将返回不受支持的服务操作结果。例如，Amazon S3 支持其资源子集的 `aws:ResourceTag`。要查看 Amazon S3 中支持 `aws:ResourceTag` 条件键的所有资源类型，请参阅服务授权参考中的 Amazon S3 定义的资源类型。

例如，假设您希望允许团队成员查看使用键值对 `team=BumbleBee` 标记的特定资源的详细信息。同时假设 Amazon Lambda 允许您标记资源，但不支持 `aws:ResourceTag` 条件键。要在此标签存在的情况下允许查看 Amazon App Mesh 和 Amazon Backup 的操作，请使用 `aws:ResourceTag` 条件键。对于 Lambda，请使用资源命名约定，其中包含团队名称作为前缀。然后包含一个单独的语句，以允许查看具有该命名约定的资源。

```
{
```

```
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "AllowViewSupported",
            "Effect": "Allow",
            "Action": [
                "appmesh:DescribeMesh",
                "backup:GetBackupPlan"
                ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:ResourceTag/team": "BumbleBee"
                }
            }
        },
        {
            "Sid": "AllowViewUnsupported",
            "Effect": "Allow",
            "Action": "lambda:GetFunction",
            "Resource": "arn:aws:lambda:*:123456789012:function:team-BumbleBee*"
        }
    ]
}
```

> **Warning**
>
> 请勿将带有 "Effect": "Allow" 的 Not 版本条件运算符用为此结果的解决方法。这些条件运算符提供否定匹配。这意味着在评估条件后，结果将被否定。在前面的示例中，在带有 StringNotEquals 运算符的 AllowViewSupported 语句中包括 lambda:GetFunction 操作将始终允许操作，无论资源是否被标记。
>
> 请勿将...IfExists版本的条件运算符用作此结果的解决方法。这表示“当键存在于请求上下文中且值匹配时，允许操作。否则，允许操作。”在前面的示例中，在带有 StringEqualsIfExists 运算符的 AllowViewSupported 语句中包含 lambda:GetFunction 操作将始终允许该操作。对于该操作，密钥不存在于上下文中，并且每次尝试查看该资源类型时都会被允许，无论资源是否被标记。

相关术语

- 全局条件键

- IAM JSON 策略元素：条件运算符
- 条件元素
- JSON 策略概述

建议 — 使用不支持的标签条件键允许服务的 NotAction

问题代码：ALLOW_NOTACTION_WITH_UNSUPPORTED_TAG_CONDITION_KEY_FOR_SERVICE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Allow NotAction with unsupported tag condition key for service: Using the effect Allow
 with NotAction and the tag condition key {{conditionKeyName}} allows only service
 actions that support the condition key. The condition key doesn't apply to some
 service actions. We recommend that you use Action instead of NotAction.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "Using the effect Allow with NotAction and the tag condition key
 {{conditionKeyName}} allows only service actions that support the condition key. The
 condition key doesn't apply to some service actions. We recommend that you use Action
 instead of NotAction."
```

解决建议

在带有 `NotAction` 和 `"Effect": "Allow"` 元素的策略的 `Condition` 元素使用不受支持的标签条件键不会影响策略授予的权限。对于不支持条件键的服务操作，该条件将被忽略。Amazon 会建议您重写逻辑以允许操作列表。

如果您将 `aws:ResourceTag` 条件键与 `NotAction` 搭配使用，则不会允许任何不支持密钥的新或现有服务操作。Amazon 会建议您明确列出要允许的操作。IAM Access Analyzer 为列出的不支持 `aws:ResourceTag` 条件键的操作返回单独的结果。有关更多信息，请参阅 建议 — 使用不支持的标签条件键允许服务。

当服务支持 `aws:ResourceTag` 条件键时，您可以使用标签来控制对该服务资源的访问。这称为基于属性的访问控制 (ABAC)。不支持这些密钥的服务需要您使用基于资源的访问控制 (RBAC) 来控制对资源的访问。

相关术语

- 全局条件键
- 比较 ABAC 与 RBAC
- IAM JSON 策略元素：条件运算符
- 条件元素
- JSON 策略概述

建议 — 服务主体的推荐条件密钥

问题代码：RECOMMENDED_CONDITION_KEY_FOR_SERVICE_PRINCIPAL

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Recommended condition key for service principal: To restrict access to the service
 principal {{servicePrincipalPrefix}} operating on your behalf, we recommend
 aws:SourceArn, aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of
 {{key}}.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "To restrict access to the service principal
 {{servicePrincipalPrefix}} operating on your behalf, we recommend aws:SourceArn,
 aws:SourceAccount, aws:SourceOrgID, or aws:SourceOrgPaths instead of {{key}}."
```

解决建议

您可以使用服务主体指定基于资源策略的 `Principal` 元素中的 Amazon Web Services 服务，这是服务的标识符。授予服务主体访问权限时，您应该使用 `aws:SourceArn`、`aws:SourceAccount`、`aws:SourceOrgID` 或 `aws:SourceOrgPaths` 条件键，而不是 `aws:Referer` 等其他条件键。这样可以帮助您防止称为混淆代理问题的安全问题。

相关术语

- Amazon Web Services 服务 主体

- Amazon 全局条件密钥：aws:SourceAccount
- Amazon 全局条件密钥：aws:SourceArn
- Amazon 全局条件键：aws:SourceOrgId
- Amazon 全局条件键：aws:SourceOrgPaths
- 混淆代理人问题

建议 — 策略中的条件密钥无关

问题代码：IRELEVANT_CONDION_KEY_IN_POLICY

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Irrelevant condition key in policy: The condition key {{condition-key}} is not relevant
 for the {{resource-type}} policy.  Use this key in an identity-based policy to govern
 access to this resource.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The condition key {{condition-key}} is not relevant for the
 {{resource-type}} policy.  Use this key in an identity-based policy to govern access
 to this resource."
```

解决建议

与基于资源的策略不相关的条件密钥。例如，`s3:ResourceAccount` 条件密钥与附加到 Amazon S3 存储桶或 Amazon S3 访问点资源类型的基于资源的策略无关。

您可以在基于身份的策略中使用条件，以便控制对资源的访问。

相关术语

- 基于身份的策略和基于资源的策略

建议 – 由于条件中存在通配符，导致存在冗余键

问题代码：REDUNDANT_KEY_DUE_TO_WILDCARD_IN_CONDITION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant key due to wildcard in condition: The key {{key}} in the condition block of
 {{operator}} is redundant because it is always matched. Remove this key to simplify
 the condition.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The key {{key}} in the condition block of {{operator}} is redundant
 because it is always matched. Remove this key to simplify the condition."
```

解决建议

从您的策略中移除冗余的条件键。由于采用了通配符模式，键始终匹配，因此这一操作就变得多余了。通过移除此键来简化条件块，同时保持相同的有效权限。

相关术语

- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥

建议 - 角色信任政策中的冗余主体

问题代码：REDUNDANT_PRINCIPAL_IN_ROLE_TRUST_POLICY

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant principal in role trust policy: The assumed-role principal
 {{redundant_principal}} is redundant with its parent role {{parent_role}}. Remove the
 assumed-role principal.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The assumed-role principal {{redundant_principal}} is redundant with
 its parent role {{parent_role}}. Remove the assumed-role principal."
```

解决建议

如果在策略的 Principal 元素中同时指定代入角色的主体及其父角色，则不会允许或拒绝任何不同的权限。例如，使用以下格式指定 Principal 元素是多余的：

```
"Principal": {
            "AWS": [
            "arn:aws:iam::AWS-account-ID:role/rolename",
            "arn:aws:iam::AWS-account-ID:assumed-role/rolename/rolesessionname"
        ]
```

我们建议删除代入角色的主体。

相关术语

- 角色会话主体

建议 – 由于条件中存在通配符，导致存在冗余语句

问题代码：REDUNDANT_STATEMENT_DUE_TO_WILDCARD_IN_CONDITION

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Redundant statement due to wildcard in condition: The key {{key}} in the condition
 block of {{operator}} does not match any values. Remove this key to simplify the
 condition.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The key {{key}} in the condition block of {{operator}} does not
 match any values. Remove this key to simplify the condition."
```

解决建议

移除不匹配任何值的条件键。此键造成了一个永远无法满足无法实现的条件，使其变得多余。请移除此键以清理您的策略，从而提高可读性和性能。

相关术语

- IAM 策略元素：条件
- 具有多个上下文键或值的条件
- Amazon 全局条件上下文密钥

建议 - 确认受众声明类型

问题代码：CONFIRM_AUDIENCE_CLAIM_TYPE

调查发现类型：SUGGESTION

调查发现详细信息

在 Amazon Web Services 管理控制台 中，此检查的结果包括以下消息：

```
Confirm audience claim type: The "{{key}}" ({{audienceType}}) claim key identifies
 the recipients that the JSON web token is intended for. Because this claim is single-
valued, do not use a qualifier.
```

在对 Amazon CLI 或者 Amazon API 的编程调用中，此检查的结果包括以下消息：

```
"findingDetails": "The "{{key}}" ({{audienceType}}) claim key identifies the recipients
 that the JSON web token is intended for. Because this claim is single-valued, do not
 use a qualifier."
```

解决建议

aud（受众）声明键是应用程序的唯一标识符，该键在您向 IdP 注册应用程序时颁发给您，并标识 JSON Web 令牌的预期接收者。受众声明可以是多值的，也可以是单值的。如果声明是多值的，请使用 `ForAllValues` 或 `ForAnyValue` 条件集运算符。如果声明是单值的，请不要使用条件集运算符。

相关术语

- 创建用于 Web 身份或 OpenID Connect 联合身份验证的角色（控制台）
- 多值上下文键
- 单值和多值条件键

使用 IAM Access Analyzer 自定义策略检查来验证策略

您可以使用自定义策略检查，根据安全标准检查新的访问。每次检查新的访问都会产生费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

使用自定义策略检查验证策略（控制台）

作为一个可选步骤，在 IAM 控制台的 JSON 策略编辑器中编辑策略时，您可以运行自定义策略检查。您可以检查与现有版本相比，更新后的策略是否授予新的访问权限。

要在编辑 IAM JSON 策略时检查新的访问

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择策略。
3. 在策略列表中，选择要编辑的策略的策略名称。您可以使用搜索框筛选策略列表。
4. 选择权限选项卡，然后选择编辑。
5. 选择 JSON 选项并更新您的策略。
6. 在策略下方的策略验证窗格中，选择检查新访问选项卡，然后选择检查策略。如果修改后的权限授予新的访问权限，则该语句将在策略验证窗格中突出显示。
7. 如果不打算授予新的访问权限，请更新策略声明并选择检查策略，直到检测不到新的访问。

> **Note**
>
> 每次检查新的访问都会产生费用。有关定价的更多详细信息，请参阅 IAM Access Analyzer 定价。

8. 选择下一步。
9. 在查看和保存页面上，查看此策略中定义的权限，然后选择保存更改。

使用自定义策略检查验证策略（Amazon CLI 或 API）

您可以从 Amazon CLI 或 IAM Access Analyzer API 运行 IAM Access Analyzer 自定义策略检查。

要运行 IAM Access Analyzer 自定义策略检查（Amazon CLI）

- 要检查与现有策略相比，更新后的策略是否允许新的访问，请运行以下命令：check-no-new-access

- 要检查策略是否不允许指定访问，请运行以下命令：check-access-not-granted
- 要检查资源策略是否可以授予对指定的资源类型的公共访问权限，请运行以下命令：check-no-public-access

要运行 IAM Access Analyzer 自定义策略检查（API）

- 要检查与现有策略相比，更新后的策略是否允许新的访问，请使用 CheckNoNewAccess API 操作。
- 要检查策略是否不允许指定访问，请使用 CheckAccessNotGranted API 操作。
- 要检查资源策略是否可以授予对指定的资源类型的公共访问权限，请使用 CheckNoPublicAccess API 操作。

IAM Access Analyzer 策略生成

作为管理员或开发人员，您可以向 IAM 实体（用户或角色）授予超出其需要的权限。IAM 提供了多个选项来帮助您优化授予的权限。其中一种选择是根据实体的访问活动生成 IAM 策略。IAM 访问分析器会查看您的 Amazon CloudTrail 日志并生成一个策略模板，其中包含角色在指定日期范围内使用的权限。您可以使用模板创建具有精细权限的策略，该策略仅授予支持特定使用案例所需的权限。

主题

- 策略生成的工作原理
- 服务和操作级别的信息
- 生成策略需知信息
- 生成策略所需的权限
- 基于 CloudTrail 活动生成策略（控制台）
- 使用其他账户中的 Amazon CloudTrail 数据生成策略
- 基于 CloudTrail 活动生成策略（Amazon CLI）
- 基于 CloudTrail 活动生成策略（Amazon API）
- IAM Access Analyzer 策略生成服务

策略生成的工作原理

IAM Access Analyzer 分析 CloudTrail 事件以识别 IAM 实体（用户或角色）已使用的操作和服务。然后，它会生成基于该活动的 IAM 策略。将附加到实体的广泛权限策略替换为生成的策略时，来可以优化实体的权限。以下是策略生成过程的高度概述。

- 设置策略模板生成 – 您可以为 IAM Access Analyzer 指定一个最多 90 天的时间段来分析历史 Amazon CloudTrail CloudTrail 事件。您必须指定一个现有服务角色或创建一个新的服务角色。服务角色允许 IAM Access Analyzer 访问您的 CloudTrail 跟踪和服务上次访问的信息，以识别所使用的服务和操作。在生成策略之前，必须指定记录账户事件的 CloudTrail 跟踪。有关 CloudTrail 数据 IAM Access Analyzer 配额的更多信息，请参阅 IAM Access Analyzer 配额。
- 生成策略 –IAM Access Analyzer 会根据 CloudTrail 事件中的访问活动生成策略。
- 查看和自定义策略 –生成策略后，您可以查看实体在指定日期范围内使用的服务和操作。您可以通过添加或删除权限、指定资源以及向策略模板添加条件来进一步自定义策略。
- 创建和附加策略 – 您可以通过创建托管策略来保存生成的策略。您可以将创建的策略附加到其活动用于生成策略的用户或角色。

服务和操作级别的信息

IAM Access Analyzer 生成 IAM 策略时，会返回信息来帮助您进一步自定义策略。生成策略时可以返回两类信息：

- 包含操作级别信息的策略 - 对于某些 Amazon 服务（例如 Amazon EC2），IAM Access Analyzer 可以识别在 CloudTrail 事件中发现的操作，并列出其生成的策略中所使用的操作。有关支持的服务的列表，请参阅 IAM Access Analyzer 策略生成服务。对于某些服务，IAM Access Analyzer 会提示您将服务的操作添加到生成的策略中。
- 包含服务级别信息的策略– IAM Access Analyzer 使用上次访问的信息创建策略模板，其中包含最近使过用的所有服务。使用 Amazon Web Services 管理控制台 时，我们会提示您查看服务并添加操作以完成策略。

有关每项服务中的操作列表，请参阅《服务授权参考》中的 Amazon 服务的操作、资源和条件键。

生成策略需知信息

在生成策略之前，请查看以下重要详细信息。

- 启用 CloudTrail 跟踪 – 您必须为账户启用 CloudTrail 跟踪，才能基于访问活动生成策略。创建 CloudTrail 跟踪时，CloudTrail 会将与跟踪相关的事件发送到您指定的 Amazon S3 存储桶。要了解如何创建 CloudTrail 跟踪，请参阅《Amazon CloudTrail 用户指南》中的为您的 Amazon 账户创建跟踪。
- 数据事件不可用 — IAM Access Analyzer 不会在生成的策略中识别数据事件（例如 Amazon S3 数据事件）的操作级别活动。

- PassRole – CloudTrail 不会跟踪 `iam:PassRole` 操作，生成的策略中也不包含该操作。
- 缩短策略生成时间 – 要加快生成策略的速度，请缩短在设置策略生成时指定的日期范围。
- 使用 CloudTrail 进行审计 – 请勿将策略生成用于审计目的；而是使用 CloudTrail 进行审计。有关使用 CloudTrail 的更多信息，请参阅使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。
- 拒绝的操作 – 策略生成会审查所有 CloudTrail 事件，包括拒绝的操作。
- 一个策略 IAM 控制台 – 您可以在 IAM 控制台中一次生成一个策略。
- 所生成策略的可用性 IAM 控制台 – 您可以在生成策略后的 7 天内在 IAM 控制台中查看生成的策略。7 天后，您必须生成新的策略。
- 策略生成配额 – 有关 IAM Access Analyzer 策略生成配额的更多信息，请参阅 IAM Access Analyzer 配额。
- 适用 Amazon S3 Standard 费率 - 当您使用策略生成功能时，IAM 访问分析器会审核 S3 存储桶中的 CloudTrail 日志。访问您的 CloudTrail 日志以生成策略不会产生额外的存储费用。Amazon 对存储在 S3 存储桶中的 CloudTrail 日志的请求和数据传输收取标准 Amazon S3 费率。
- Amazon Control Tower 支持 – 策略生成不支持使用 Amazon Control Tower 在生成策略时创建的 Amazon CloudTrail 跟踪，原因如下：
  - 来自组织的 CloudTrail 数据记录在另一个账户中，即 Amazon Control Tower 日志存档账户。
  - 由于 Amazon Control Tower 的服务控制策略 (SCP) 对 S3 日志记录存储桶设置的限制，因此无法重新配置存储这些日志的 S3 存储桶的权限。

生成策略所需的权限

首次生成策略所需的权限与为后续使用生成策略所需的权限不同。有关更多信息，请参阅 Amazon Identity and Access Management Access Analyzer 入门。

首次设置

首次生成策略时，必须在账户中选择一个合适的现有服务角色或创建一个新的服务角色。服务角色允许 IAM Access Analyzer 访问账户中的 CloudTrail 以及服务上次访问的信息。仅管理员有权创建和配置角色。因此，我们建议管理员在首次设置时创建服务角色。要了解有关创建服务角色所需权限的更多信息，请参阅创建角色以向 Amazon 服务委派权限。

创建服务角色所需的权限

创建服务角色时，您需要为该角色配置两个策略。您可以将 IAM 权限策略附加到角色，它指定角色可以执行的操作。您还可以将角色信任策略附加到角色，它指定可以使用该角色的主体。

第一个示例策略显示了生成策略所需的服务角色的权限策略。第二个示例策略显示了服务角色所需的角色信任策略。在使用 Amazon API 或 Amazon CLI 生成策略时，可以使用这些策略帮助创建服务角色。在策略生成过程中使用 IAM 控制台创建服务角色时，我们会为您生成这些策略。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "cloudtrail:GetTrail",
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetServiceLastAccessedDetails",
                "iam:GenerateServiceLastAccessedDetails"
            ],
            "Resource": "*"
        },
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject",
                "s3:ListBucket"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket",
                "arn:aws:s3:::amzn-s3-demo-bucket/*"
            ]
        }
    ]
}
```

以下示例策略显示的角色信任策略具有允许 IAM Access Analyzer 担任角色的权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "access-analyzer.amazonaws.com"
      },
      "Action": "sts:AssumeRole"
    }
  ]
}
```

后续使用

要在 Amazon Web Services 管理控制台 中生成策略，IAM 用户必须拥有允许他们将用于生成策略的服务角色传递给 IAM Access Analyzer 的权限策略。`iam:PassRole` 通常与 `iam:GetRole` 共同使用，使用户能够获取准备进行传递的角色的详细信息。在此示例中，用户只能传递位于指定账户中并且名称以 `AccessAnalyzerMonitorServiceRole*` 开头的角色。要了解将 IAM 角色传递给 Amazon 服务的更多信息，请参阅向用户授予权限以将角色传递给 Amazon 服务。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowUserToPassRole",
      "Effect": "Allow",
      "Action": [
        "iam:GetRole",
        "iam:PassRole"
      ],
      "Resource": "arn:aws:iam::123456789012:role/service-role/
AccessAnalyzerMonitorServiceRole*"
    }
  ]
```

```
}
```

您还必须具有以下 IAM Access Analyzer 权限才能在 Amazon Web Services 管理控制台、Amazon API 中或 Amazon CLI 中生成策略，如以下策略语句所示。

```
{
  "Sid": "AllowUserToGeneratePolicy",
  "Effect": "Allow",
  "Action": [
    "access-analyzer:CancelPolicyGeneration",
    "access-analyzer:GetGeneratedPolicy",
    "access-analyzer:ListPolicyGenerations",
    "access-analyzer:StartPolicyGeneration"
  ],
  "Resource": "*"
}
```

首次使用和后续使用

使用 Amazon Web Services 管理控制台 生成策略时，必须授予 `cloudtrail:ListTrails` 在账户中列出 CloudTrail 跟踪的权限，如以下策略语句所示。

```
{
  "Sid": "AllowUserToListTrails",
  "Effect": "Allow",
  "Action": [
    "CloudTrail:ListTrails"
  ],
  "Resource": "*"
}
```

基于 CloudTrail 活动生成策略（控制台）

您可以为 IAM 用户或角色生成策略。

步骤 1：基于 CloudTrail 活动生成策略

以下过程介绍了如何使用 Amazon Web Services 管理控制台 为角色生成策略。

为 IAM 角色生成策略

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在左侧的导航窗格中，选择角色。

   > **Note**
   >
   > 基于 IAM 用户的活动生成策略的步骤几乎完全相同。要执行此操作，请选择 Users (用户)，不要选择 Roles (角色)。

3. 在账户的角色列表中，选择要将其活动用于生成策略的角色的名称。
4. 在 Permissions (权限) 选项卡的基于 CloudTrail 事件生成策略部分，选择 Generate policy (生成策略)。
5. 在 Generate policy (生成策略)页面，指定希望 IAM Access Analyzer 分析 CloudTrail 事件（以对角色采取操作）的时间段。最多可以选择 90 天。我们建议您选择尽可能短的时间段以缩短策略生成时间。
6. 在 CloudTrail 访问部分中，选择合适的现有角色或创建新角色（如果不存在合适角色）。此角色授予 IAM Access Analyzer 代表您访问 CloudTrail 数据的权限，以查看访问活动并识别已使用的服务和操作。要了解该角色所需的权限，请参阅 生成策略所需的权限。
7. 在要分析的 CloudTrail 跟踪部分，指定记录账户事件的 CloudTrail 跟踪。

   如果您选择将日志存储在其他账户中的 CloudTrail 跟踪记录，则会显示一个有关跨账户访问的信息框。跨账户访问需要额外的设置。要了解更多信息，请稍后参阅本主题中的 Choose a role for cross-account access。
8. 选择 Generate Policy (生成策略)。
9. 在策略生成过程中，您将返回到 Permissions（权限）选项卡上的 Roles（角色）Summary（摘要）页面。等到 Policy request details (策略请求详细信息) 部分中的状态显示为 Success (成功)后，选择 View generated policy (查看生成的策略)。您可以在策略生成后的七天内进行查看。如果生成另一个策略，则会将现有策略替换为生成的新策略。

步骤 2：查看权限并为所使用的服务添加操作

查看 IAM Access Analyzer 识别到的角色所使用的服务和操作。您可以为生成的策略模板添加用于任何服务的操作。

1. 查看以下部分：

- 在 Review permissions (查看权限) 页面，查看 Actions included in the generated policy (生成的策略中包含的操作) 列表。该列表显示了角色在指定日期范围内使用的 IAM Access Analyzer 已识别到的服务和操作。
- Services used (使用的服务) 部分显示了角色在指定日期范围内使用的 IAM Access Analyzer 已识别到的其他服务。有关使用了哪些操作的信息可能不适用于本部分中列出的服务。使用列出的各项服务的菜单手动选择您希望在策略中包含的操作。

2. 添加完操作后，选择 Next (下一步)。

步骤 3：进一步自定义生成的策略

您可以通过添加或删除权限或指定资源来进一步自定义策略。

自定义生成的策略

1. 请更新策略模板。策略模板包含支持资源级权限的操作的资源 ARN 占位符，如下图所示。资源级权限指的是能够指定允许用户对哪些资源执行操作的能力。我们建议您使用 ARN 在策略中为支持资源级权限的操作指定个人资源。您可以将占位符资源 ARN 替换为使用案例的有效资源 ARN。

   如果操作不支持资源级权限，则必须使用通配符 (*) 来指定所有资源都可能受到操作的影响。要了解哪些 Amazon 服务支持资源级权限，请参阅 Amazon services that work with IAM（能够与 IAM 搭配使用的亚马逊云科技服务）。有关各项服务中的操作列表，以及要了解哪些操作支持资源级权限，请参阅 Amazon 服务的操作、资源和条件键。

2. （可选）在模板中添加、修改或删除 JSON 策略语句。要了解有关编写 JSON 策略的更多信息，请参阅创建 IAM 策略（控制台）。
3. 自定义策略模板后，可以选择以下选项：
   - （可选）您可以复制模板中的 JSON，以便在 Generated policy (生成的策略) 页面之外单独使用。例如，您想要使用 JSON 在其他账户中创建策略时。如果模板中策略的字符超过 JSON 策略的字符上限 6144，则会将该策略拆分为多个策略。
   - 选择 Next (下一步)，以在同一账户中查看并创建托管策略。

步骤 4：查看并创建托管策略

如果您有权创建和附加 IAM 策略，则可以通过生成的策略创建托管策略。然后，您可以将策略附加到账户中的用户或角色。

查看并创建策略

1. 在 Review and create managed policy (查看并创建托管策略) 页面，为创建的策略输入 Name (名称) 和 Description (描述)（可选）。
2. （可选）在 Summary (摘要) 部分，您可以查看策略中将包含的权限。
3. （可选）通过以密钥值对的形式附加标签来向策略添加元数据。有关将在 IAM 中使用标签的更多信息，请参阅 Tagging IAM resources（标记 IAM 资源）。
4. 完成后，请执行以下操作之一：
   - 您可以将新策略直接附加到用于生成策略的角色。为此，请在页面底部附近，选中将策略附加到 ***YourRoleName*** 旁边的复选框。然后选择 Create and attach policy (创建并附加策略)。
   - 否则，请选择 Create policy (创建策略)。您创建的策略位于 IAM 控制台上，可以在 Policies (策略) 导航窗格的策略列表中找到它。
5. 您可以将创建的策略附加到账户中的实体。附加策略后，您可以删除可以附加到该实体的任何其他过于宽泛的策略。要了解如何附加托管策略，请参阅添加 IAM 身份权限（控制台）。

使用其他账户中的 Amazon CloudTrail 数据生成策略

您可以创建将数据存储在中央账户中的 CloudTrail 跟踪记录，以简化管理活动。例如，您可以使用 Amazon Organizations 创建跟踪记录，以为该企业中的所有 Amazon Web Services 账户 记录所有事件。跟踪记录属于一个中央账户。如果要为账户中的用户或角色生成策略，而该账户与您存储 CloudTrail 日志数据所在的账户不同，则必须授予跨账户访问权限。为此，您需要一个角色和一个存储

桶策略来授予 IAM Access Analyzer 对您的 CloudTrail 日志的权限。有关创建 Organizations 跟踪记录的更多信息，请参阅为企业创建跟踪记录。

在此示例中，假定您要为账户 A 中的用户或角色生成策略。账户 A 中的 CloudTrail 跟踪记录将 CloudTrail 日志存储在账户 B 的存储桶中。在生成策略之前，您必须进行以下更新：

1. 选择一个现有角色，或创建一个新的服务角色，以授予 IAM Access Analyzer 对账户 B（存储您的 CloudTrail 日志的位置）中存储桶的访问权限。
2. 验证账户 B 中的 Amazon S3 存储桶对象所有权和存储桶权限策略，以便 IAM 访问分析器可以访问存储桶中的对象。

步骤 1：选择或创建用于跨账户访问的角色

- 在 Generate policy（生成策略）屏幕上，如果您的账户中存在具有所需权限的角色，则会为您预先选择 Use an existing role（使用现有角色）的选项。否则，请选择 Create and use a new service role（创建和使用新的服务角色）。新角色用于向 IAM Access Analyzer 授予对账户 B 中 CloudTrail 日志的访问权限。

步骤 2：验证或更新账户 B 中的 Amazon S3 存储桶配置

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 Amazon S3 控制台：https://console.aws.amazon.com/s3/。
2. 在 Buckets（存储桶）列表中，请选择存储 CloudTrail 日志的存储桶名称。
3. 请选择 Permissions（权限）选项卡，然后转到 Object Ownership（对象所有权）部分。

   使用 Amazon S3 对象所有权存储桶设置来控制上传到存储桶的对象的所有权。默认情况下，当其他 Amazon Web Services 账户 将对象上传到存储桶时，这些对象仍为上传账户所有。要生成策略，存储桶中的所有对象都必须归存储桶拥有者所有。您可能需要更改您存储桶的 Object Ownership（对象所有权）设置，具体视您的 ACL 使用案例而定。将 Object Ownership（对象所有权）设置为以下选项之一。

   - 强制存储桶拥有者（推荐）
   - 首选存储桶拥有者

> **Important**
>
> 要成功生成策略，存储桶中的对象必须归存储桶拥有者所有。如果您选择使用 Bucket owner preferred（首选存储桶拥有者），您只能在对象所有权更改后的时间段生成策略。

如需了解关于 Amazon S3 中的对象所有权的更多信息，请参阅《Amazon S3 用户指南》中的控制存储桶的对象所有权和禁用 ACL。

4. 向账户 B 中的 Amazon S3 存储桶策略添加权限，以允许对账户 A 中的角色进行访问。

以下示例策略针对名为 `amzn-s3-demo-bucket` 的存储桶允许 `ListBucket` 和 `GetObject`。如果访问存储桶的角色属于您企业中的账户并且名称以 `AccessAnalyzerMonitorServiceRole` 开头，则其会允许访问权限。使用 `aws:PrincipalArn` 作为 `Resource` 元素中的 `Condition` 可确保角色只能访问账户活动（如果其属于账户 A）。您可以将 `amzn-s3-demo-bucket` 替换为存储桶名称，将 `optional-prefix` 替换为存储桶的可选前缀，将 `organization-id` 替换为组织 ID。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PolicyGenerationBucketPolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "s3:GetObject",
        "s3:ListBucket"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/optional-prefix/
AWSLogs/organization-id/${aws:PrincipalAccount}/*"
      ],
      "Condition": {
```

```
        "StringEquals": {
          "aws:PrincipalOrgID": "organization-id"
        },
        "ArnLike": {
          "aws:PrincipalArn": "arn:aws:iam:::${aws:PrincipalAccount}:role/
service-role/AccessAnalyzerMonitorServiceRole*"
        }
      }
    }
  ]
}
```

5. 如果您使用 Amazon KMS 加密日志，请在您存储 CloudTrail 日志的账户中更新您的 Amazon KMS 密钥策略，以授予 IAM Access Analyzer 访问权限来使用密钥，如以下策略示例所示。将 `CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN` 替换为您跟踪记录的 ARN，将 `organization-id` 替换为您的企业 ID。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": "kms:Decrypt",
      "Resource": "*",
      "Condition": {
        "StringEquals": {
          "kms:EncryptionContext:aws:cloudtrail:arn":
 "CROSS_ACCOUNT_ORG_TRAIL_FULL_ARN",
          "aws:PrincipalOrgID": "organization-id"
        },
        "StringLike": {
          "kms:ViaService": [
            "access-analyzer.*.amazonaws.com",
            "s3.*.amazonaws.com"
            ]
        },
        "ArnLike": {
```

```
          "aws:PrincipalArn": "arn:aws:iam:::${aws:PrincipalAccount}:role/
service-role/AccessAnalyzerMonitorServiceRole*"
        }
      }
     }
    ]
}
```

基于 CloudTrail 活动生成策略（Amazon CLI）

您可以通过 Amazon CLI 使用以下命令生成策略。

生成策略

- aws accessanalyzer start-policy-generation

查看生成的策略

- aws accessanalyzer get-generated-policy

取消策略生成请求

- aws accessanalyzer cancel-policy-generation

查看策略生成请求列表

- aws accessanalyzer list-policy-generations

基于 CloudTrail 活动生成策略（Amazon API）

您可以通过 Amazon API 使用以下操作生成策略。

生成策略

- StartPolicyGeneration

查看生成的策略

- GetGeneratedPolicy

取消策略生成请求

- CancelPolicyGeneration

查看策略生成请求列表

- ListPolicyGenerations

IAM Access Analyzer 策略生成服务

下表列出了 IAM Access Analyzer 将为其生成包含操作级别信息的策略的 Amazon 服务。有关每项服务中的操作列表，请参阅《服务授权参考》中的 Amazon 服务的操作、资源和条件键。

Amazon 以 JSON 格式提供策略生成信息，以简化策略管理工作流程的自动化。借助服务参考信息，您可以从机器可读文件中跨 Amazon Web Services 服务 获取策略生成信息。有关更多信息，请参阅《服务授权参考》中的用于编程访问的简化 Amazon Web Services 服务信息。

服务	服务前缀
Amazon Identity and Access Management Access Analyzer	access-analyzer
Amazon 账户管理	账户
Amazon Certificate Manager	acm
Amazon Managed Workflows for Apache Airflow	airflow
Amazon Amplify	amplify
Amazon Amplify UI Builder	amplifyuibuilder
Amazon AppIntegrations	app-integrations
Amazon AppConfig	appconfig

服务	服务前缀
Amazon AppFlow	appflow
Amazon Application Cost Profiler	application-cost-profiler
Amazon CloudWatch Application Insights	applicationinsights
Amazon App Mesh	appmesh
Amazon WorkSpaces Applications	appstream
Amazon AppSync	appsync
Amazon Managed Service for Prometheus	aps
Amazon Athena	athena
Amazon Audit Manager	auditmanager
Amazon Auto Scaling	自动扩缩
Amazon Web Services Marketplace	aws-marketplace
Amazon Backup	备份
Amazon Batch	批处理
Amazon Braket	Braket
Amazon Budgets	预算
Amazon Cloud9	cloud9
Amazon CloudFormation	cloudformation
Amazon CloudFront	cloudfront
Amazon CloudHSM	cloudhsm
Amazon CloudSearch	cloudsearch

服务	服务前缀
Amazon CloudTrail	cloudtrail
Amazon CloudWatch	cloudwatch
Amazon CodeArtifact	codeartifact
Amazon CodeDeploy	codedeploy
Amazon CodeGuru Profiler	codeguru-profiler
Amazon CodeGuru Reviewer	codeguru-reviewer
Amazon CodePipeline	codepipeline
AWS CodeStar	codestar
AWS CodeStar 通知	codestar-notifications
Amazon Cognito 身份	cognito-identity
Amazon Cognito 用户群体	cognito-idp
Amazon Cognito Sync	cognito-sync
Amazon Comprehend Medical	comprehendmedical
Amazon Compute Optimizer	compute-optimizer
Amazon Config	config
Amazon Connect Customer	connect
Amazon 成本和使用情况报告	cur
Amazon Glue DataBrew	databrew
Amazon Web Services Data Exchange	dataexchange

服务	服务前缀
Amazon Data Pipeline	datapipeline
DynamoDB Accelerator	dax
Amazon Device Farm	devicefarm
Amazon DevOps Guru	devops-guru
Amazon Direct Connect	directconnect
Amazon Data Lifecycle Manager	dlm
Amazon Database Migration Service	dms
Amazon DocumentDB Elastic Clusters	docdb-elastic
Amazon DynamoDB	dynamodb
Amazon Elastic Block Store	ebs
Amazon Elastic Compute Cloud	ec2
Amazon Elastic Container Registry	ecr
Amazon Elastic Container Registry Public	ecr-public
Amazon Elastic Container Service	ecs
Amazon Elastic Kubernetes Service	eks
Amazon ElastiCache	elasticache
Amazon Elastic Beanstalk	elasticbeanstalk
Amazon Elastic File System	elasticfilesystem
Elastic Load Balancing	elasticloadbalancing
Amazon Elastic Transcoder	elastictranscoder

服务	服务前缀
Amazon EMR 在 EKS 上（EMR 容器）	emr-containers
Amazon EMR Serverless	emr-serverless
Amazon OpenSearch Service	es
Amazon EventBridge	events
Amazon CloudWatch Evidently	evidently
Amazon FinSpace	finspace
Amazon Data Firehose	Firehose
Amazon Fault Injection Service	fis
Amazon Firewall Manager	fms
Amazon Fraud Detector	frauddetector
Amazon FSx	fsx
Amazon GameLift Servers	GameLift
Amazon Location Service	geo
Amazon Glacier	glacier
Amazon Managed Grafana	grafana
Amazon IoT Greengrass	greengrass
Amazon Ground Station	groundstation
Amazon GuardDuty	guardduty
Amazon HealthLake	healthlake
Amazon Honeycode	honeycode

服务	服务前缀
Amazon Identity and Access Management	IAM
Amazon 身份存储	identitystore
EC2 Image Builder	imagebuilder
Amazon Inspector Classic	inspector
Amazon Inspector	inspector2
Amazon IoT	iot
Amazon IoT Analytics	iotanalytics
Amazon IoT Core Device Advisor	iotdeviceadvisor
Amazon IoT Events	iotevents
Amazon IoT Fleet Hub	iotfleethub
Amazon IoT SiteWise	iotsitewise
Amazon IoT TwinMaker	iottwinmaker
Amazon IoT Wireless	iotwireless
Amazon Interactive Video Service	ivs
Amazon Interactive Video Service Chat	ivschat
Amazon Managed Streaming for Apache Kafka	kafka
Amazon Managed Streaming for Kafka Connect	kafkaconnect
Amazon Kendra	kendra
Amazon Kinesis	kinesis
Amazon Kinesis Analytics V2	kinesisanalytics

服务	服务前缀
Amazon Key Management Service	kms
Amazon Lambda	lambda
Amazon Lex	Lex
Amazon License Manager Linux Subscriptions Manager	license-manager-linux-subscriptions
Amazon Lightsail	lightsail
Amazon CloudWatch Logs	logs
Amazon Lookout for Equipment	lookoutequipment
Amazon Lookout for Metrics	lookoutmetrics
Amazon Lookout for Vision	lookoutvision
Amazon Mainframe Modernization	m2
Amazon Managed Blockchain	managedblockchain
AWS Elemental MediaConnect	mediaconnect
AWS Elemental MediaConvert	mediaconvert
AWS Elemental MediaLive	medialive
AWS Elemental MediaStore	mediastore
AWS Elemental MediaTailor	mediatailor
Amazon MemoryDB	memorydb
Amazon Transform MGN	mgn
Amazon Migration Hub	mgh

服务	服务前缀
Amazon Migration Hub 策略建议	migration hub-strategy
Amazon Pinpoint	mobiletargeting
Amazon MQ	mq
Amazon Network Manager	networkmanager
Amazon Nimble Studio	nimble
Amazon HealthOmics	omics
Amazon OpsWorks	opsworks
Amazon OpsWorks CM	opsworks-cm
Amazon Outposts	outposts
Amazon Organizations	组织
Amazon Panorama	panorama
Amazon 性能详情	pi
Amazon EventBridge Pipes	pipes
Amazon Polly	polly
Amazon Connect Customer Profiles	配置文件
Amazon QLDB	qldb
Amazon Resource Access Manager	ram
Amazon 回收站	rbin
Amazon Relational Database Service	rds
Amazon Redshift	redshift

服务	服务前缀
Amazon Redshift Data API	redshift-data
Amazon Migration Hub Refactor Spaces	refactor-spaces
Amazon Rekognition	rekognition
Amazon Resilience Hub	resiliencehub
Amazon 资源探索器	resource-explorer-2
Amazon Resource Groups	resource-groups
Amazon RoboMaker	robomaker
Amazon Identity and Access Management Roles Anywhere	rolesanywhere
Amazon Route 53	route53
Amazon Route 53 Recovery 控制	route53-recovery-control-config
Amazon Route 53 Recovery 就绪性	route53-recovery-readiness
Amazon Route 53 Resolver	route53resolver
Amazon CloudWatch RUM	rum
Amazon Simple Storage Service	S3
Amazon S3 on Outposts	s3-outposts
Amazon SageMaker AI 地理空间功能	sagemaker-geospatial
Savings Plans	savingsplans
Amazon EventBridge Schemas	schemas

服务	服务前缀
Amazon SimpleDB	sdb
Amazon Secrets Manager	secretsmanager
Amazon Security Hub CSPM	securityhub
Amazon Security Lake	securitylake
Amazon Serverless Application Repository	serverlessrepo
Amazon Service Catalog	servicecatalog
Amazon Cloud Map	servicediscovery
服务配额	servicequotas
Amazon Simple Email Service	ses
Amazon Shield	shield
Amazon Signer	signer
Amazon SimSpace Weaver	simspaceweaver
Amazon Server Migration Service	sms
Amazon Pinpoint 短信和语音服务	sms-voice
Amazon Snowball Edge	snowball
Amazon Simple Queue Service	sqs
Amazon Systems Manager	ssm
Amazon Systems Manager Incident Manager	ssm-incidents
适用于 SAP 的 Amazon Systems Manager	ssm-sap
Amazon Step Functions	states

服务	服务前缀
Amazon Security Token Service	sts
Amazon Simple Workflow Service	swf
Amazon CloudWatch Synthetics	synthetics
Amazon Resource Groups Tagging API	标签
Amazon Textract	textract
Amazon Timestream	timestream
Amazon 电信网络生成器	tnb
Amazon Transcribe	transcribe
Amazon Transfer Family	转移
Amazon Translate	translate
Connect Customer Voice ID	voiceid
Amazon VPC Lattice	vpc-lattice
Amazon WAFV2	wafv2
Amazon Well-Architected Tool	wellarchitected
Amazon Connect Customer Wisdom	wisdom
Amazon WorkLink	worklink
Amazon WorkSpaces	工作区
Amazon X-Ray	xray

策略生成信息的操作

下表列出了 IAM Access Analyzer 将为其生成包含操作级别信息的策略的操作。

> **Important**
>
> `iam:UpdateAccountName` 操作将于 2026 年 4 月 22 日弃用。2026 年 4 月 22 日之后，只有 `account:PutAccountName` 权限才能控制账户名称更新访问权限。我们强烈建议更新任何控制账户名称更新的服务控制策略 (SCP)，以使用 `account:PutAccountName` 权限。

服务前缀	操作
access-analyzer	access-analyzer:ApplyArchiveRule
	access-analyzer:CancelPolicyGeneration
	access-analyzer:CheckAccessNotGranted
	access-analyzer:CheckNoNewAccess
	access-analyzer:CheckNoPublicAccess
	access-analyzer:CreateAccessPreview
	access-analyzer:CreateAnalyzer
	access-analyzer:CreateArchiveRule
	access-analyzer:DeleteAnalyzer
	access-analyzer:DeleteArchiveRule
	access-analyzer:GenerateFindingRecommendation
	access-analyzer:GetAccessPreview
	access-analyzer:GetAnalyzedResource
	access-analyzer:GetAnalyzer
	access-analyzer:GetArchiveRule
	access-analyzer:GetFinding
	access-analyzer:GetFindingRecommendation

服务前缀	操作
	access-analyzer:GetFindingsStatistics
	access-analyzer:GetGeneratedPolicy
	access-analyzer:ListAccessPreviewFindings
	access-analyzer:ListAccessPreviews
	access-analyzer:ListAnalyzedResources
	access-analyzer:ListAnalyzers
	access-analyzer:ListArchiveRules
	access-analyzer:ListFindings
	access-analyzer:ListPolicyGenerations
	access-analyzer:StartPolicyGeneration
	access-analyzer:StartResourceScan
	access-analyzer:UpdateAnalyzer
	access-analyzer:UpdateArchiveRule
	access-analyzer:UpdateFindings
	access-analyzer:ValidatePolicy

服务前缀	操作
账户	account:AcceptPrimaryEmailUpdate account:DeleteAlternateContact account:DisableRegion account:EnableRegion account:GetAccountInformation account:GetAlternateContact account:GetContactInformation account:GetGovCloudAccountInformation account:GetPrimaryEmail account:GetRegionOptStatus account:ListRegions account:PutAccountName account:PutAlternateContact account:PutContactInformation account:StartPrimaryEmailUpdate

服务前缀	操作
acm	acm:DeleteCertificate acm:DescribeCertificate acm:ExportCertificate acm:GetAccountConfiguration acm:GetCertificate acm:ImportCertificate acm:ListCertificates acm:PutAccountConfiguration acm:RenewCertificate acm:RequestCertificate acm:ResendValidationEmail acm:UpdateCertificateOptions
airflow	airflow:CreateCliToken airflow:CreateEnvironment airflow:CreateWebLoginToken airflow:DeleteEnvironment airflow:GetEnvironment airflow:ListEnvironments airflow:PublishMetrics airflow:UpdateEnvironment

服务前缀	操作
amplify	amplify:CreateApp
	amplify:CreateBackendEnvironment
	amplify:CreateBranch
	amplify:CreateDeployment
	amplify:CreateDomainAssociation
	amplify:CreateWebHook
	amplify:DeleteApp
	amplify:DeleteBackendEnvironment
	amplify:DeleteBranch
	amplify:DeleteDomainAssociation
	amplify:DeleteJob
	amplify:DeleteWebHook
	amplify:GenerateAccessLogs
	amplify:GetApp
	amplify:GetArtifactUrl
	amplify:GetBackendEnvironment
	amplify:GetBranch
	amplify:GetDomainAssociation
	amplify:GetJob
	amplify:GetWebHook
	amplify:ListApps

服务前缀	操作
	amplify:ListArtifacts
	amplify:ListBackendEnvironments
	amplify:ListBranches
	amplify:ListDomainAssociations
	amplify:ListJobs
	amplify:ListWebHooks
	amplify:StartDeployment
	amplify:StartJob
	amplify:StopJob
	amplify:UpdateApp
	amplify:UpdateBranch
	amplify:UpdateDomainAssociation
	amplify:UpdateWebHook

服务前缀	操作
amplifyuibuilder	amplifyuibuilder:CreateComponent amplifyuibuilder:CreateForm amplifyuibuilder:CreateTheme amplifyuibuilder:DeleteComponent amplifyuibuilder:DeleteForm amplifyuibuilder:DeleteTheme amplifyuibuilder:ExportComponents amplifyuibuilder:ExportThemes amplifyuibuilder:GetCodegenJob amplifyuibuilder:ListCodegenJobs amplifyuibuilder:ListComponents amplifyuibuilder:ListForms amplifyuibuilder:ListThemes amplifyuibuilder:ResetMetadataFlag amplifyuibuilder:StartCodegenJob amplifyuibuilder:UpdateComponent amplifyuibuilder:UpdateForm amplifyuibuilder:UpdateTheme

服务前缀	操作
app-integrations	app-integrations:CreateApplication app-integrations:CreateDataIntegration app-integrations:CreateDataIntegrationAssociation app-integrations:CreateEventIntegration app-integrations:DeleteApplication app-integrations:DeleteDataIntegration app-integrations:DeleteEventIntegration app-integrations:GetApplication app-integrations:GetDataIntegration app-integrations:GetEventIntegration app-integrations:ListApplicationAssociations app-integrations:ListApplications app-integrations:ListDataIntegrationAssociations app-integrations:ListDataIntegrations app-integrations:ListEventIntegrationAssociations app-integrations:ListEventIntegrations app-integrations:UpdateApplication app-integrations:UpdateDataIntegration app-integrations:UpdateDataIntegrationAssociation app-integrations:UpdateEventIntegration

服务前缀	操作
appconfig	appconfig:CreateApplication appconfig:CreateConfigurationProfile appconfig:CreateDeploymentStrategy appconfig:CreateEnvironment appconfig:CreateExtension appconfig:CreateExtensionAssociation appconfig:CreateHostedConfigurationVersion appconfig:DeleteApplication appconfig:DeleteConfigurationProfile appconfig:DeleteDeploymentStrategy appconfig:DeleteEnvironment appconfig:DeleteExtension appconfig:DeleteExtensionAssociation appconfig:DeleteHostedConfigurationVersion appconfig:GetAccountSettings appconfig:GetApplication appconfig:GetConfiguration appconfig:GetConfigurationProfile appconfig:GetDeployment appconfig:GetDeploymentStrategy appconfig:GetEnvironment

服务前缀	操作
	appconfig:GetExtension appconfig:GetExtensionAssociation appconfig:GetHostedConfigurationVersion appconfig:ListApplications appconfig:ListConfigurationProfiles appconfig:ListDeploymentStrategies appconfig:ListDeployments appconfig:ListEnvironments appconfig:ListExtensionAssociations appconfig:ListExtensions appconfig:ListHostedConfigurationVersions appconfig:StartDeployment appconfig:StopDeployment appconfig:UpdateAccountSettings appconfig:UpdateApplication appconfig:UpdateConfigurationProfile appconfig:UpdateDeploymentStrategy appconfig:UpdateEnvironment appconfig:UpdateExtension appconfig:UpdateExtensionAssociation appconfig:ValidateConfiguration

服务前缀	操作
appflow	appflow:CancelFlowExecutions
	appflow:CreateConnectorProfile
	appflow:CreateFlow
	appflow:DeleteConnectorProfile
	appflow:DeleteFlow
	appflow:DescribeConnector
	appflow:DescribeConnectorEntity
	appflow:DescribeConnectorProfiles
	appflow:DescribeConnectors
	appflow:DescribeFlow
	appflow:DescribeFlowExecutionRecords
	appflow:ListConnectorEntities
	appflow:ListConnectors
	appflow:ListFlows
	appflow:RegisterConnector
	appflow:ResetConnectorMetadataCache
	appflow:StartFlow
	appflow:StopFlow
	appflow:UnRegisterConnector
	appflow:UpdateConnectorProfile
	appflow:UpdateConnectorRegistration

服务前缀	操作
	appflow:UpdateFlow

服务前缀	操作
applicationinsights	applicationinsights:AddWorkload applicationinsights:CreateApplication applicationinsights:CreateComponent applicationinsights:CreateLogPattern applicationinsights:DeleteApplication applicationinsights:DeleteComponent applicationinsights:DeleteLogPattern applicationinsights:DescribeApplication applicationinsights:DescribeComponent applicationinsights:DescribeComponentConfiguration applicationinsights:DescribeComponentConfigurationRecommendation applicationinsights:DescribeLogPattern applicationinsights:DescribeObservation applicationinsights:DescribeProblem applicationinsights:DescribeProblemObservations applicationinsights:DescribeWorkload applicationinsights:ListApplications applicationinsights:ListComponents applicationinsights:ListConfigurationHistory applicationinsights:ListLogPatternSets

服务前缀	操作
	applicationinsights:ListLogPatterns applicationinsights:ListProblems applicationinsights:ListWorkloads applicationinsights:RemoveWorkload applicationinsights:UpdateApplication applicationinsights:UpdateComponent applicationinsights:UpdateComponentConfiguration applicationinsights:UpdateLogPattern applicationinsights:UpdateWorkload

服务前缀	操作
appmesh	appmesh:CreateGatewayRoute
	appmesh:CreateMesh
	appmesh:CreateRoute
	appmesh:CreateVirtualGateway
	appmesh:CreateVirtualNode
	appmesh:CreateVirtualRouter
	appmesh:CreateVirtualService
	appmesh:DeleteGatewayRoute
	appmesh:DeleteMesh
	appmesh:DeleteRoute
	appmesh:DeleteVirtualGateway
	appmesh:DeleteVirtualNode
	appmesh:DeleteVirtualRouter
	appmesh:DeleteVirtualService
	appmesh:DescribeGatewayRoute
	appmesh:DescribeMesh
	appmesh:DescribeRoute
	appmesh:DescribeVirtualGateway
	appmesh:DescribeVirtualNode
	appmesh:DescribeVirtualRouter
	appmesh:DescribeVirtualService

服务前缀	操作
	appmesh:ListGatewayRoutes
	appmesh:ListMeshes
	appmesh:ListRoutes
	appmesh:ListVirtualGateways
	appmesh:ListVirtualNodes
	appmesh:ListVirtualRouters
	appmesh:ListVirtualServices
	appmesh:StreamAggregatedResources
	appmesh:UpdateGatewayRoute
	appmesh:UpdateMesh
	appmesh:UpdateRoute
	appmesh:UpdateVirtualGateway
	appmesh:UpdateVirtualNode
	appmesh:UpdateVirtualRouter
	appmesh:UpdateVirtualService

服务前缀	操作
appstream	appstream:AssociateAppBlockBuilderAppBlock appstream:AssociateApplicationFleet appstream:AssociateApplicationToEntitlement appstream:AssociateFleet appstream:AssociateSoftwareToImageBuilder appstream:BatchAssociateUserStack appstream:BatchDisassociateUserStack appstream:CopyImage appstream:CreateAppBlock appstream:CreateAppBlockBuilder appstream:CreateAppBlockBuilderStreamingURL appstream:CreateApplication appstream:CreateDirectoryConfig appstream:CreateEntitlement appstream:CreateFleet appstream:CreateImageBuilder appstream:CreateImageBuilderStreamingURL appstream:CreateStack appstream:CreateStreamingURL appstream:CreateThemeForStack appstream:CreateUpdatedImage

服务前缀	操作
	appstream:CreateUsageReportSubscription appstream:CreateUser appstream:DeleteAppBlock appstream:DeleteAppBlockBuilder appstream:DeleteApplication appstream:DeleteDirectoryConfig appstream:DeleteEntitlement appstream:DeleteFleet appstream:DeleteImage appstream:DeleteImageBuilder appstream:DeleteImagePermissions appstream:DeleteStack appstream:DeleteThemeForStack appstream:DeleteUsageReportSubscription appstream:DeleteUser appstream:DescribeAppBlockBuilderAppBlockAssociations appstream:DescribeAppBlockBuilders appstream:DescribeAppBlocks appstream:DescribeAppLicenseUsage appstream:DescribeApplicationFleetAssociations appstream:DescribeApplications

服务前缀	操作
	appstream:DescribeDirectoryConfigs appstream:DescribeEntitlements appstream:DescribeFleets appstream:DescribeImageBuilders appstream:DescribeImagePermissions appstream:DescribeImages appstream:DescribeSessions appstream:DescribeStacks appstream:DescribeThemeForStack appstream:DescribeUsageReportSubscriptions appstream:DescribeUserStackAssociations appstream:DescribeUsers appstream:DisableUser appstream:DisassociateAppBlockBuilderAppBlock appstream:DisassociateApplicationFleet appstream:DisassociateApplicationFromEntitlement appstream:DisassociateFleet appstream:DisassociateSoftwareFromImageBuilder appstream:EnableUser appstream:ExpireSession appstream:GetExportImageTask

服务前缀	操作
	appstream:ListAssociatedFleets appstream:ListAssociatedStacks appstream:ListEntitledApplications appstream:ListExportImageTasks appstream:StartAppBlockBuilder appstream:StartFleet appstream:StartImageBuilder appstream:StartSoftwareDeploymentToImageBuilder appstream:StopAppBlockBuilder appstream:StopFleet appstream:StopImageBuilder appstream:UpdateAppBlockBuilder appstream:UpdateApplication appstream:UpdateDirectoryConfig appstream:UpdateEntitlement appstream:UpdateFleet appstream:UpdateImagePermissions appstream:UpdateStack appstream:UpdateThemeForStack

服务前缀	操作
appsync	appsync:AssociateApi
	appsync:AssociateMergedGraphqlApi
	appsync:AssociateSourceGraphqlApi
	appsync:CreateApi
	appsync:CreateApiCache
	appsync:CreateApiKey
	appsync:CreateChannelNamespace
	appsync:CreateDataSource
	appsync:CreateDomainName
	appsync:CreateFunction
	appsync:CreateGraphqlApi
	appsync:CreateResolver
	appsync:CreateType
	appsync:DeleteApi
	appsync:DeleteApiCache
	appsync:DeleteApiKey
	appsync:DeleteChannelNamespace
	appsync:DeleteDataSource
	appsync:DeleteDomainName
	appsync:DeleteFunction
	appsync:DeleteGraphqlApi

服务前缀	操作
	appsync:DeleteResolver
	appsync:DeleteType
	appsync:DisassociateApi
	appsync:DisassociateMergedGraphqlApi
	appsync:DisassociateSourceGraphqlApi
	appsync:EvaluateCode
	appsync:EvaluateMappingTemplate
	appsync:FlushApiCache
	appsync:GetApi
	appsync:GetApiAssociation
	appsync:GetApiCache
	appsync:GetChannelNamespace
	appsync:GetDataSource
	appsync:GetDataSourceIntrospection
	appsync:GetDomainName
	appsync:GetFunction
	appsync:GetGraphqlApi
	appsync:GetGraphqlApiEnvironmentVariables
	appsync:GetIntrospectionSchema
	appsync:GetResolver
	appsync:GetSchemaCreationStatus

服务前缀	操作
	appsync:GetSourceApiAssociation
	appsync:GetType
	appsync:ListApiKeys
	appsync:ListApis
	appsync:ListChannelNamespaces
	appsync:ListDataSources
	appsync:ListDomainNames
	appsync:ListFunctions
	appsync:ListGraphqlApis
	appsync:ListResolvers
	appsync:ListResolversByFunction
	appsync:ListSourceApiAssociations
	appsync:ListTypes
	appsync:ListTypesByAssociation
	appsync:PutGraphqlApiEnvironmentVariables
	appsync:StartDataSourceIntrospection
	appsync:StartSchemaCreation
	appsync:StartSchemaMerge
	appsync:UpdateApi
	appsync:UpdateApiCache
	appsync:UpdateApiKey

服务前缀	操作
	appsync:UpdateChannelNamespace appsync:UpdateDataSource appsync:UpdateDomainName appsync:UpdateFunction appsync:UpdateGraphqlApi appsync:UpdateResolver appsync:UpdateSourceApiAssociation appsync:UpdateType

服务前缀	操作
aps	aps:CreateAlertManagerDefinition aps:CreateAnomalyDetector aps:CreateLoggingConfiguration aps:CreateQueryLoggingConfiguration aps:CreateRuleGroupsNamespace aps:CreateWorkspace aps:DeleteAlertManagerDefinition aps:DeleteAnomalyDetector aps:DeleteLoggingConfiguration aps:DeleteQueryLoggingConfiguration aps:DeleteResourcePolicy aps:DeleteRuleGroupsNamespace aps:DeleteScraper aps:DeleteScraperLoggingConfiguration aps:DeleteWorkspace aps:DescribeAlertManagerDefinition aps:DescribeAnomalyDetector aps:DescribeLoggingConfiguration aps:DescribeQueryLoggingConfiguration aps:DescribeResourcePolicy aps:DescribeRuleGroupsNamespace

服务前缀	操作
	aps:DescribeScraper aps:DescribeScraperLoggingConfiguration aps:DescribeWorkspace aps:DescribeWorkspaceConfiguration aps:GetDefaultScraperConfiguration aps:ListAnomalyDetectors aps:ListRuleGroupsNamespaces aps:ListScrapers aps:ListWorkspaces aps:PutAlertManagerDefinition aps:PutAnomalyDetector aps:PutResourcePolicy aps:PutRuleGroupsNamespace aps:UpdateLoggingConfiguration aps:UpdateQueryLoggingConfiguration aps:UpdateScraper aps:UpdateScraperLoggingConfiguration aps:UpdateWorkspaceAlias aps:UpdateWorkspaceConfiguration

服务前缀	操作
athena	athena:BatchGetNamedQuery
	athena:BatchGetPreparedStatement
	athena:BatchGetQueryExecution
	athena:CancelCapacityReservation
	athena:CreateCapacityReservation
	athena:CreateDataCatalog
	athena:CreateNamedQuery
	athena:CreateNotebook
	athena:CreatePreparedStatement
	athena:CreatePresignedNotebookUrl
	athena:CreateWorkGroup
	athena:DeleteCapacityReservation
	athena:DeleteDataCatalog
	athena:DeleteNamedQuery
	athena:DeleteNotebook
	athena:DeletePreparedStatement
	athena:DeleteWorkGroup
	athena:ExportNotebook
	athena:GetCalculationExecution
	athena:GetCalculationExecutionCode
	athena:GetCalculationExecutionStatus

服务前缀	操作
	athena:GetCapacityAssignmentConfiguration
	athena:GetCapacityReservation
	athena:GetDataCatalog
	athena:GetDatabase
	athena:GetNamedQuery
	athena:GetNotebookMetadata
	athena:GetPreparedStatement
	athena:GetQueryExecution
	athena:GetQueryResults
	athena:GetQueryResultsStream
	athena:GetQueryRuntimeStatistics
	athena:GetResourceDashboard
	athena:GetSession
	athena:GetSessionEndpoint
	athena:GetSessionStatus
	athena:GetTableMetadata
	athena:GetWorkGroup
	athena:ImportNotebook
	athena:ListApplicationDPUSizes
	athena:ListCalculationExecutions
	athena:ListCapacityReservations

服务前缀	操作
	athena:ListDataCatalogs
	athena:ListDatabases
	athena:ListEngineVersions
	athena:ListExecutors
	athena:ListNamedQueries
	athena:ListNotebookMetadata
	athena:ListNotebookSessions
	athena:ListPreparedStatements
	athena:ListQueryExecutions
	athena:ListSessions
	athena:ListTableMetadata
	athena:ListWorkGroups
	athena:PutCapacityAssignmentConfiguration
	athena:StartCalculationExecution
	athena:StartQueryExecution
	athena:StartSession
	athena:StopCalculationExecution
	athena:StopQueryExecution
	athena:TerminateSession
	athena:UpdateCapacityReservation
	athena:UpdateDataCatalog

服务前缀	操作
	athena:UpdateNamedQuery
	athena:UpdateNotebook
	athena:UpdateNotebookMetadata
	athena:UpdatePreparedStatement
	athena:UpdateWorkGroup

服务前缀	操作
auditmanager	auditmanager:AssociateAssessmentReportEvidenceFolder auditmanager:BatchAssociateAssessmentReportEvidence auditmanager:BatchCreateDelegationByAssessment auditmanager:BatchDeleteDelegationByAssessment auditmanager:BatchDisassociateAssessmentReportEvidence auditmanager:BatchImportEvidenceToAssessmentControl auditmanager:CreateAssessment auditmanager:CreateAssessmentFramework auditmanager:CreateAssessmentReport auditmanager:CreateControl auditmanager:DeleteAssessment auditmanager:DeleteAssessmentFramework auditmanager:DeleteAssessmentFrameworkShare auditmanager:DeleteAssessmentReport auditmanager:DeleteControl auditmanager:DeregisterAccount auditmanager:DeregisterOrganizationAdminAccount auditmanager:DisassociateAssessmentReportEvidenceFolder auditmanager:GetAccountStatus auditmanager:GetAssessment auditmanager:GetAssessmentFramework

服务前缀	操作
	auditmanager:GetAssessmentReportUrl auditmanager:GetChangeLogs auditmanager:GetControl auditmanager:GetDelegations auditmanager:GetEvidence auditmanager:GetEvidenceByEvidenceFolder auditmanager:GetEvidenceFileUploadUrl auditmanager:GetEvidenceFolder auditmanager:GetEvidenceFoldersByAssessment auditmanager:GetEvidenceFoldersByAssessmentControl auditmanager:GetInsights auditmanager:GetInsightsByAssessment auditmanager:GetOrganizationAdminAccount auditmanager:GetServicesInScope auditmanager:GetSettings auditmanager:ListAssessmentControlInsightsByControlDomain auditmanager:ListAssessmentFrameworkShareRequests auditmanager:ListAssessmentFrameworks auditmanager:ListAssessmentReports auditmanager:ListAssessments auditmanager:ListControlDomainInsights

服务前缀	操作
	auditmanager:ListControlDomainInsightsByAssessment auditmanager:ListControlInsightsByControlDomain auditmanager:ListControls auditmanager:ListKeywordsForDataSource auditmanager:ListNotifications auditmanager:RegisterAccount auditmanager:RegisterOrganizationAdminAccount auditmanager:StartAssessmentFrameworkShare auditmanager:UpdateAssessment auditmanager:UpdateAssessmentControl auditmanager:UpdateAssessmentControlSetStatus auditmanager:UpdateAssessmentFramework auditmanager:UpdateAssessmentFrameworkShare auditmanager:UpdateAssessmentStatus auditmanager:UpdateControl auditmanager:UpdateSettings auditmanager:ValidateAssessmentReportIntegrity

服务前缀	操作
自动扩缩	autoscaling:AttachInstances
	autoscaling:AttachLoadBalancerTargetGroups
	autoscaling:AttachLoadBalancers
	autoscaling:AttachTrafficSources
	autoscaling:BatchDeleteScheduledAction
	autoscaling:BatchPutScheduledUpdateGroupAction
	autoscaling:CancelInstanceRefresh
	autoscaling:CompleteLifecycleAction
	autoscaling:CreateAutoScalingGroup
	autoscaling:CreateLaunchConfiguration
	autoscaling:DeleteAutoScalingGroup
	autoscaling:DeleteLaunchConfiguration
	autoscaling:DeleteLifecycleHook
	autoscaling:DeleteNotificationConfiguration
	autoscaling:DeletePolicy
	autoscaling:DeleteScheduledAction
	autoscaling:DeleteWarmPool
	autoscaling:DescribeAccountLimits
	autoscaling:DescribeAdjustmentTypes
	autoscaling:DescribeAutoScalingGroups
	autoscaling:DescribeAutoScalingInstances

服务前缀	操作
	autoscaling:DescribeAutoScalingNotificationTypes
	autoscaling:DescribeInstanceRefreshes
	autoscaling:DescribeLaunchConfigurations
	autoscaling:DescribeLifecycleHookTypes
	autoscaling:DescribeLifecycleHooks
	autoscaling:DescribeLoadBalancerTargetGroups
	autoscaling:DescribeLoadBalancers
	autoscaling:DescribeMetricCollectionTypes
	autoscaling:DescribeNotificationConfigurations
	autoscaling:DescribePolicies
	autoscaling:DescribeScalingActivities
	autoscaling:DescribeScalingProcessTypes
	autoscaling:DescribeScheduledActions
	autoscaling:DescribeTerminationPolicyTypes
	autoscaling:DescribeTrafficSources
	autoscaling:DescribeWarmPool
	autoscaling:DetachInstances
	autoscaling:DetachLoadBalancerTargetGroups
	autoscaling:DetachLoadBalancers
	autoscaling:DetachTrafficSources
	autoscaling:DisableMetricsCollection

服务前缀	操作
	autoscaling:EnableMetricsCollection autoscaling:EnterStandby autoscaling:ExecutePolicy autoscaling:ExitStandby autoscaling:GetPredictiveScalingForecast autoscaling:PutLifecycleHook autoscaling:PutNotificationConfiguration autoscaling:PutScalingPolicy autoscaling:PutScheduledUpdateGroupAction autoscaling:PutWarmPool autoscaling:RecordLifecycleActionHeartbeat autoscaling:ResumeProcesses autoscaling:RollbackInstanceRefresh autoscaling:SetDesiredCapacity autoscaling:SetInstanceHealth autoscaling:SetInstanceProtection autoscaling:StartInstanceRefresh autoscaling:SuspendProcesses autoscaling:TerminateInstanceInAutoScalingGroup autoscaling:UpdateAutoScalingGroup
aws-marketplace	aws-marketplace:GetEntitlements

服务前缀	操作
备份	backup:CancelLegalHold
	backup:CreateBackupPlan
	backup:CreateBackupSelection
	backup:CreateBackupVault
	backup:CreateFramework
	backup:CreateLegalHold
	backup:CreateLogicallyAirGappedBackupVault
	backup:CreateReportPlan
	backup:CreateRestoreAccessBackupVault
	backup:CreateRestoreTestingPlan
	backup:CreateRestoreTestingSelection
	backup:CreateTieringConfiguration
	backup:DeleteBackupPlan
	backup:DeleteBackupSelection
	backup:DeleteBackupVault
	backup:DeleteBackupVaultAccessPolicy
	backup:DeleteBackupVaultLockConfiguration
	backup:DeleteBackupVaultNotifications
	backup:DeleteFramework
	backup:DeleteRecoveryPoint
	backup:DeleteReportPlan

服务前缀	操作
	backup:DeleteRestoreTestingPlan
	backup:DeleteRestoreTestingSelection
	backup:DeleteTieringConfiguration
	backup:DescribeBackupJob
	backup:DescribeBackupVault
	backup:DescribeCopyJob
	backup:DescribeFramework
	backup:DescribeGlobalSettings
	backup:DescribeProtectedResource
	backup:DescribeRecoveryPoint
	backup:DescribeRegionSettings
	backup:DescribeReportJob
	backup:DescribeReportPlan
	backup:DescribeRestoreJob
	backup:DescribeScanJob
	backup:DisassociateRecoveryPoint
	backup:DisassociateRecoveryPointFromParent
	backup:ExportBackupPlanTemplate
	backup:GetBackupPlan
	backup:GetBackupPlanFromJSON
	backup:GetBackupPlanFromTemplate

服务前缀	操作
	backup:GetBackupSelection
	backup:GetBackupVaultAccessPolicy
	backup:GetBackupVaultNotifications
	backup:GetLegalHold
	backup:GetRecoveryPointRestoreMetadata
	backup:GetRestoreJobMetadata
	backup:GetRestoreTestingInferredMetadata
	backup:GetRestoreTestingPlan
	backup:GetRestoreTestingSelection
	backup:GetSupportedResourceTypes
	backup:GetTieringConfiguration
	backup:ListBackupJobSummaries
	backup:ListBackupJobs
	backup:ListBackupPlanTemplates
	backup:ListBackupPlanVersions
	backup:ListBackupPlans
	backup:ListBackupSelections
	backup:ListBackupVaults
	backup:ListCopyJobSummaries
	backup:ListCopyJobs
	backup:ListFrameworks

服务前缀	操作
	backup:ListIndexedRecoveryPoints
	backup:ListLegalHolds
	backup:ListProtectedResources
	backup:ListRecoveryPointsByBackupVault
	backup:ListRecoveryPointsByLegalHold
	backup:ListRecoveryPointsByResource
	backup:ListReportJobs
	backup:ListReportPlans
	backup:ListRestoreAccessBackupVaults
	backup:ListRestoreJobSummaries
	backup:ListRestoreJobs
	backup:ListRestoreJobsByProtectedResource
	backup:ListRestoreTestingPlans
	backup:ListRestoreTestingSelections
	backup:ListScanJobSummaries
	backup:ListScanJobs
	backup:ListTieringConfigurations
	backup:PutBackupVaultAccessPolicy
	backup:PutBackupVaultLockConfiguration
	backup:PutBackupVaultNotifications
	backup:PutRestoreValidationResult

服务前缀	操作
	backup:StartBackupJob
	backup:StartCopyJob
	backup:StartReportJob
	backup:StartRestoreJob
	backup:StopBackupJob
	backup:UpdateBackupPlan
	backup:UpdateFramework
	backup:UpdateGlobalSettings
	backup:UpdateRecoveryPointLifecycle
	backup:UpdateRegionSettings
	backup:UpdateReportPlan
	backup:UpdateRestoreTestingPlan
	backup:UpdateRestoreTestingSelection
	backup:UpdateTieringConfiguration

服务前缀	操作
批处理	batch:CancelJob batch:CreateComputeEnvironment batch:CreateConsumableResource batch:CreateJobQueue batch:CreateSchedulingPolicy batch:CreateServiceEnvironment batch:DeleteComputeEnvironment batch:DeleteConsumableResource batch:DeleteJobQueue batch:DeleteSchedulingPolicy batch:DeleteServiceEnvironment batch:DeregisterJobDefinition batch:DescribeComputeEnvironments batch:DescribeConsumableResource batch:DescribeJobDefinitions batch:DescribeJobQueues batch:DescribeJobs batch:DescribeSchedulingPolicies batch:DescribeServiceEnvironments batch:DescribeServiceJob batch:GetJobQueueSnapshot

服务前缀	操作
	batch:ListConsumableResources batch:ListJobs batch:ListJobsByConsumableResource batch:ListSchedulingPolicies batch:ListServiceJobs batch:RegisterJobDefinition batch:SubmitJob batch:SubmitServiceJob batch:TerminateJob batch:TerminateServiceJob batch:UpdateComputeEnvironment batch:UpdateConsumableResource batch:UpdateJobQueue batch:UpdateSchedulingPolicy batch:UpdateServiceEnvironment

服务前缀	操作
Braket	braket:CancelJob braket:CancelQuantumTask braket:CreateJob braket:CreateQuantumTask braket:CreateSpendingLimit braket:GetDevice braket:GetJob braket:GetQuantumTask braket:SearchDevices braket:SearchJobs braket:SearchQuantumTasks braket:SearchSpendingLimits

服务前缀	操作
预算	budgets:CreateBudgetAction budgets:DeleteBudgetAction budgets:DescribeBudgetAction budgets:DescribeBudgetActionHistories budgets:DescribeBudgetActionsForAccount budgets:DescribeBudgetActionsForBudget budgets:ExecuteBudgetAction budgets:ModifyBudget budgets:UpdateBudgetAction budgets:ViewBudget
cloud9	cloud9:CreateEnvironmentEC2 cloud9:CreateEnvironmentMembership cloud9:DeleteEnvironment cloud9:DeleteEnvironmentMembership cloud9:DescribeEnvironmentMemberships cloud9:DescribeEnvironmentStatus cloud9:DescribeEnvironments cloud9:ListEnvironments cloud9:UpdateEnvironment cloud9:UpdateEnvironmentMembership

服务前缀	操作
cloudformation	cloudformation:BatchDescribeTypeConfigurations cloudformation:CancelUpdateStack cloudformation:ContinueUpdateRollback cloudformation:CreateChangeSet cloudformation:CreateGeneratedTemplate cloudformation:CreateStack cloudformation:CreateStackInstances cloudformation:CreateStackSet cloudformation:DeactivateType cloudformation:DeleteChangeSet cloudformation:DeleteGeneratedTemplate cloudformation:DeleteStack cloudformation:DeleteStackInstances cloudformation:DeleteStackSet cloudformation:DeregisterType cloudformation:DescribeAccountLimits cloudformation:DescribeChangeSet cloudformation:DescribeChangeSetHooks cloudformation:DescribeEvents cloudformation:DescribeGeneratedTemplate cloudformation:DescribeOrganizationsAccess

服务前缀	操作
	cloudformation:DescribePublisher cloudformation:DescribeResourceScan cloudformation:DescribeStackDriftDetectionStatus cloudformation:DescribeStackEvents cloudformation:DescribeStackInstance cloudformation:DescribeStackResource cloudformation:DescribeStackResourceDrifts cloudformation:DescribeStackResources cloudformation:DescribeStackSet cloudformation:DescribeStackSetOperation cloudformation:DescribeStacks cloudformation:DescribeType cloudformation:DescribeTypeRegistration cloudformation:DetectStackDrift cloudformation:DetectStackResourceDrift cloudformation:DetectStackSetDrift cloudformation:EstimateTemplateCost cloudformation:ExecuteChangeSet cloudformation:GetGeneratedTemplate cloudformation:GetHookResult cloudformation:GetStackPolicy

服务前缀	操作
	cloudformation:GetTemplate cloudformation:GetTemplateSummary cloudformation:ImportStacksToStackSet cloudformation:ListChangeSets cloudformation:ListExports cloudformation:ListGeneratedTemplates cloudformation:ListHookResults cloudformation:ListImports cloudformation:ListResourceScanRelatedResources cloudformation:ListResourceScanResources cloudformation:ListResourceScans cloudformation:ListStackInstanceResourceDrifts cloudformation:ListStackInstances cloudformation:ListStackRefactors cloudformation:ListStackResources cloudformation:ListStackSetAutoDeploymentTargets cloudformation:ListStackSetOperationResults cloudformation:ListStackSetOperations cloudformation:ListStackSets cloudformation:ListTypeRegistrations cloudformation:ListTypeVersions

服务前缀	操作
	cloudformation:ListTypes cloudformation:PublishType cloudformation:RecordHandlerProgress cloudformation:RegisterPublisher cloudformation:RegisterType cloudformation:RollbackStack cloudformation:SetStackPolicy cloudformation:SetTypeConfiguration cloudformation:SetTypeDefaultVersion cloudformation:SignalResource cloudformation:StartResourceScan cloudformation:StopStackSetOperation cloudformation:TestType cloudformation:UpdateGeneratedTemplate cloudformation:UpdateStack cloudformation:UpdateStackInstances cloudformation:UpdateStackSet cloudformation:UpdateTerminationProtection cloudformation:ValidateTemplate

服务前缀	操作
cloudfront	cloudfront:AssociateAlias
	cloudfront:AssociateDistributionTenantWebACL
	cloudfront:AssociateDistributionWebACL
	cloudfront:CreateCachePolicy
	cloudfront:CreateCloudFrontOriginAccessIdentity
	cloudfront:CreateConnectionFunction
	cloudfront:CreateContinuousDeploymentPolicy
	cloudfront:CreateDistributionTenant
	cloudfront:CreateFieldLevelEncryptionConfig
	cloudfront:CreateFieldLevelEncryptionProfile
	cloudfront:CreateFunction
	cloudfront:CreateInvalidation
	cloudfront:CreateKeyGroup
	cloudfront:CreateKeyValueStore
	cloudfront:CreateMonitoringSubscription
	cloudfront:CreateOriginAccessControl
	cloudfront:CreateOriginRequestPolicy
	cloudfront:CreatePublicKey
	cloudfront:CreateRealtimeLogConfig
	cloudfront:CreateResponseHeadersPolicy
	cloudfront:CreateTrustStore

服务前缀	操作
	cloudfront:DeleteAnycastIpList
	cloudfront:DeleteCachePolicy
	cloudfront:DeleteCloudFrontOriginAccessIdentity
	cloudfront:DeleteConnectionFunction
	cloudfront:DeleteConnectionGroup
	cloudfront:DeleteContinuousDeploymentPolicy
	cloudfront:DeleteDistribution
	cloudfront:DeleteDistributionTenant
	cloudfront:DeleteFieldLevelEncryptionConfig
	cloudfront:DeleteFieldLevelEncryptionProfile
	cloudfront:DeleteFunction
	cloudfront:DeleteKeyGroup
	cloudfront:DeleteKeyValueStore
	cloudfront:DeleteMonitoringSubscription
	cloudfront:DeleteOriginAccessControl
	cloudfront:DeleteOriginRequestPolicy
	cloudfront:DeletePublicKey
	cloudfront:DeleteRealtimeLogConfig
	cloudfront:DeleteResponseHeadersPolicy
	cloudfront:DeleteStreamingDistribution
	cloudfront:DeleteTrustStore

服务前缀	操作
	cloudfront:DeleteVpcOrigin
	cloudfront:DescribeFunction
	cloudfront:DescribeKeyValueStore
	cloudfront:DisassociateDistributionTenantWebACL
	cloudfront:DisassociateDistributionWebACL
	cloudfront:GetAnycastIpList
	cloudfront:GetCachePolicy
	cloudfront:GetCachePolicyConfig
	cloudfront:GetCloudFrontOriginAccessIdentity
	cloudfront:GetCloudFrontOriginAccessIdentityConfig
	cloudfront:GetContinuousDeploymentPolicy
	cloudfront:GetContinuousDeploymentPolicyConfig
	cloudfront:GetDistributionConfig
	cloudfront:GetFieldLevelEncryption
	cloudfront:GetFieldLevelEncryptionConfig
	cloudfront:GetFieldLevelEncryptionProfile
	cloudfront:GetFieldLevelEncryptionProfileConfig
	cloudfront:GetFunction
	cloudfront:GetInvalidation
	cloudfront:GetInvalidationForDistributionTenant
	cloudfront:GetKeyGroup

服务前缀	操作
	cloudfront:GetKeyGroupConfig
	cloudfront:GetMonitoringSubscription
	cloudfront:GetOriginAccessControl
	cloudfront:GetOriginAccessControlConfig
	cloudfront:GetOriginRequestPolicy
	cloudfront:GetOriginRequestPolicyConfig
	cloudfront:GetPublicKey
	cloudfront:GetPublicKeyConfig
	cloudfront:GetRealtimeLogConfig
	cloudfront:GetResponseHeadersPolicy
	cloudfront:GetResponseHeadersPolicyConfig
	cloudfront:GetStreamingDistribution
	cloudfront:GetStreamingDistributionConfig
	cloudfront:GetVpcOrigin
	cloudfront:ListAnycastIpLists
	cloudfront:ListCachePolicies
	cloudfront:ListCloudFrontOriginAccessIdentities
	cloudfront:ListConflictingAliases
	cloudfront:ListConnectionFunctions
	cloudfront:ListConnectionGroups
	cloudfront:ListContinuousDeploymentPolicies

服务前缀	操作
	cloudfront:ListDistributionTenants
	cloudfront:ListDistributionTenantsByCustomization
	cloudfront:ListDistributions
	cloudfront:ListDistributionsByAnycastIpListId
	cloudfront:ListDistributionsByCachePolicyId
	cloudfront:ListDistributionsByConnectionMode
	cloudfront:ListDistributionsByKeyGroup
	cloudfront:ListDistributionsByOriginRequestPolicyId
	cloudfront:ListDistributionsByRealtimeLogConfig
	cloudfront:ListDistributionsByResponseHeadersPolicyId
	cloudfront:ListDistributionsByVpcOriginId
	cloudfront:ListDistributionsByWebACLId
	cloudfront:ListFieldLevelEncryptionConfigs
	cloudfront:ListFieldLevelEncryptionProfiles
	cloudfront:ListFunctions
	cloudfront:ListInvalidations
	cloudfront:ListInvalidationsForDistributionTenant
	cloudfront:ListKeyGroups
	cloudfront:ListKeyValueStores
	cloudfront:ListOriginAccessControls
	cloudfront:ListOriginRequestPolicies

服务前缀	操作
	cloudfront:ListPublicKeys
	cloudfront:ListRealtimeLogConfigs
	cloudfront:ListResponseHeadersPolicies
	cloudfront:ListStreamingDistributions
	cloudfront:ListTrustStores
	cloudfront:PublishConnectionFunction
	cloudfront:PublishFunction
	cloudfront:TestConnectionFunction
	cloudfront:TestFunction
	cloudfront:UpdateAnycastIpList
	cloudfront:UpdateCachePolicy
	cloudfront:UpdateCloudFrontOriginAccessIdentity
	cloudfront:UpdateConnectionFunction
	cloudfront:UpdateConnectionGroup
	cloudfront:UpdateContinuousDeploymentPolicy
	cloudfront:UpdateDistribution
	cloudfront:UpdateDistributionTenant
	cloudfront:UpdateFieldLevelEncryptionConfig
	cloudfront:UpdateFieldLevelEncryptionProfile
	cloudfront:UpdateFunction
	cloudfront:UpdateKeyGroup

服务前缀	操作
	cloudfront:UpdateKeyValueStore cloudfront:UpdateOriginAccessControl cloudfront:UpdateOriginRequestPolicy cloudfront:UpdatePublicKey cloudfront:UpdateRealtimeLogConfig cloudfront:UpdateResponseHeadersPolicy cloudfront:UpdateTrustStore
cloudhsm	cloudhsm:CreateHsm cloudhsm:DeleteBackup cloudhsm:DeleteHsm cloudhsm:DeleteResourcePolicy cloudhsm:DescribeBackups cloudhsm:DescribeClusters cloudhsm:GetResourcePolicy cloudhsm:InitializeCluster cloudhsm:ModifyBackupAttributes cloudhsm:ModifyCluster cloudhsm:PutResourcePolicy cloudhsm:RestoreBackup

服务前缀	操作
cloudsearch	cloudsearch:BuildSuggesters
	cloudsearch:CreateDomain
	cloudsearch:DefineAnalysisScheme
	cloudsearch:DefineExpression
	cloudsearch:DefineIndexField
	cloudsearch:DefineSuggester
	cloudsearch:DeleteAnalysisScheme
	cloudsearch:DeleteDomain
	cloudsearch:DeleteExpression
	cloudsearch:DeleteIndexField
	cloudsearch:DeleteSuggester
	cloudsearch:DescribeAnalysisSchemes
	cloudsearch:DescribeAvailabilityOptions
	cloudsearch:DescribeDomainEndpointOptions
	cloudsearch:DescribeDomains
	cloudsearch:DescribeExpressions
	cloudsearch:DescribeIndexFields
	cloudsearch:DescribeScalingParameters
	cloudsearch:DescribeServiceAccessPolicies
	cloudsearch:DescribeSuggesters
	cloudsearch:IndexDocuments

服务前缀	操作
	cloudsearch:ListDomainNames
	cloudsearch:UpdateAvailabilityOptions
	cloudsearch:UpdateDomainEndpointOptions
	cloudsearch:UpdateScalingParameters
	cloudsearch:UpdateServiceAccessPolicies

服务前缀	操作
cloudtrail	cloudtrail:CancelQuery cloudtrail:CreateChannel cloudtrail:CreateDashboard cloudtrail:CreateEventDataStore cloudtrail:CreateTrail cloudtrail:DeleteChannel cloudtrail:DeleteDashboard cloudtrail:DeleteEventDataStore cloudtrail:DeleteResourcePolicy cloudtrail:DeleteTrail cloudtrail:DeregisterOrganizationDelegatedAdmin cloudtrail:DescribeQuery cloudtrail:DescribeTrails cloudtrail:DisableFederation cloudtrail:GenerateQuery cloudtrail:GetChannel cloudtrail:GetDashboard cloudtrail:GetEventConfiguration cloudtrail:GetEventDataStore cloudtrail:GetEventDataStoreData cloudtrail:GetEventSelectors

服务前缀	操作
	cloudtrail:GetImport cloudtrail:GetInsightSelectors cloudtrail:GetResourcePolicy cloudtrail:GetTrail cloudtrail:GetTrailStatus cloudtrail:ListChannels cloudtrail:ListDashboards cloudtrail:ListEventDataStores cloudtrail:ListImportFailures cloudtrail:ListImports cloudtrail:ListInsightsData cloudtrail:ListPublicKeys cloudtrail:ListQueries cloudtrail:ListTrails cloudtrail:LookupEvents cloudtrail:PutEventConfiguration cloudtrail:PutEventSelectors cloudtrail:PutInsightSelectors cloudtrail:PutResourcePolicy cloudtrail:RegisterOrganizationDelegatedAdmin cloudtrail:RestoreEventDataStore

服务前缀	操作
	cloudtrail:SearchSampleQueries cloudtrail:StartEventDataStoreIngestion cloudtrail:StartImport cloudtrail:StartLogging cloudtrail:StartQuery cloudtrail:StopEventDataStoreIngestion cloudtrail:StopImport cloudtrail:StopLogging cloudtrail:UpdateChannel cloudtrail:UpdateDashboard cloudtrail:UpdateEventDataStore cloudtrail:UpdateTrail

服务前缀	操作
cloudwatch	cloudwatch:DeleteAlarms
	cloudwatch:DeleteAnomalyDetector
	cloudwatch:DeleteDashboards
	cloudwatch:DeleteInsightRules
	cloudwatch:DeleteMetricStream
	cloudwatch:DescribeAlarmHistory
	cloudwatch:DescribeAlarms
	cloudwatch:DescribeAlarmsForMetric
	cloudwatch:DescribeAnomalyDetectors
	cloudwatch:DescribeInsightRules
	cloudwatch:DisableAlarmActions
	cloudwatch:DisableInsightRules
	cloudwatch:EnableAlarmActions
	cloudwatch:EnableInsightRules
	cloudwatch:GetDashboard
	cloudwatch:GetInsightRuleReport
	cloudwatch:GetMetricStatistics
	cloudwatch:GetMetricStream
	cloudwatch:ListDashboards
	cloudwatch:ListManagedInsightRules
	cloudwatch:ListMetricStreams

服务前缀	操作
	cloudwatch:PutAnomalyDetector
	cloudwatch:PutCompositeAlarm
	cloudwatch:PutDashboard
	cloudwatch:PutInsightRule
	cloudwatch:PutManagedInsightRules
	cloudwatch:PutMetricAlarm
	cloudwatch:PutMetricStream
	cloudwatch:SetAlarmState
	cloudwatch:StartMetricStreams
	cloudwatch:StopMetricStreams

服务前缀	操作
codeartifact	codeartifact:AssociateExternalConnection codeartifact:CopyPackageVersions codeartifact:CreateDomain codeartifact:CreateRepository codeartifact:DeleteDomain codeartifact:DeleteDomainPermissionsPolicy codeartifact:DeletePackage codeartifact:DeletePackageVersions codeartifact:DeleteRepository codeartifact:DeleteRepositoryPermissionsPolicy codeartifact:DescribeDomain codeartifact:DescribePackage codeartifact:DescribePackageVersion codeartifact:DescribeRepository codeartifact:DisassociateExternalConnection codeartifact:DisposePackageVersions codeartifact:GetAssociatedPackageGroup codeartifact:GetAuthorizationToken codeartifact:GetDomainPermissionsPolicy codeartifact:GetPackageVersionAsset codeartifact:GetPackageVersionReadme

服务前缀	操作
	codeartifact:GetRepositoryEndpoint codeartifact:GetRepositoryPermissionsPolicy codeartifact:ListDomains codeartifact:ListPackageGroups codeartifact:ListPackageVersionAssets codeartifact:ListPackageVersionDependencies codeartifact:ListPackageVersions codeartifact:ListPackages codeartifact:ListRepositories codeartifact:ListRepositoriesInDomain codeartifact:PublishPackageVersion codeartifact:PutDomainPermissionsPolicy codeartifact:PutPackageMetadata codeartifact:PutPackageOriginConfiguration codeartifact:PutRepositoryPermissionsPolicy codeartifact:ReadFromRepository codeartifact:UpdatePackageVersionsStatus codeartifact:UpdateRepository

服务前缀	操作
codedeploy	codedeploy:BatchGetApplicationRevisions
	codedeploy:BatchGetApplications
	codedeploy:BatchGetDeploymentGroups
	codedeploy:BatchGetDeploymentInstances
	codedeploy:BatchGetDeploymentTargets
	codedeploy:BatchGetDeployments
	codedeploy:BatchGetOnPremisesInstances
	codedeploy:ContinueDeployment
	codedeploy:CreateApplication
	codedeploy:CreateDeployment
	codedeploy:CreateDeploymentConfig
	codedeploy:CreateDeploymentGroup
	codedeploy:DeleteApplication
	codedeploy:DeleteDeploymentConfig
	codedeploy:DeleteDeploymentGroup
	codedeploy:DeleteGitHubAccountToken
	codedeploy:DeleteResourcesByExternalId
	codedeploy:DeregisterOnPremisesInstance
	codedeploy:GetApplication
	codedeploy:GetApplicationRevision
	codedeploy:GetDeployment

服务前缀	操作
	codedeploy:GetDeploymentConfig
	codedeploy:GetDeploymentGroup
	codedeploy:GetDeploymentInstance
	codedeploy:GetDeploymentTarget
	codedeploy:GetOnPremisesInstance
	codedeploy:ListApplicationRevisions
	codedeploy:ListApplications
	codedeploy:ListDeploymentConfigs
	codedeploy:ListDeploymentGroups
	codedeploy:ListDeploymentInstances
	codedeploy:ListDeploymentTargets
	codedeploy:ListDeployments
	codedeploy:ListGitHubAccountTokenNames
	codedeploy:ListOnPremisesInstances
	codedeploy:PutLifecycleEventHookExecutionStatus
	codedeploy:RegisterApplicationRevision
	codedeploy:RegisterOnPremisesInstance
	codedeploy:SkipWaitTimeForInstanceTermination
	codedeploy:StopDeployment
	codedeploy:UpdateApplication
	codedeploy:UpdateDeploymentGroup

服务前缀	操作
codeguru-profiler	codeguru-profiler:AddNotificationChannels codeguru-profiler:BatchGetFrameMetricData codeguru-profiler:CreateProfilingGroup codeguru-profiler:DeleteProfilingGroup codeguru-profiler:DescribeProfilingGroup codeguru-profiler:GetFindingsReportAccountSummary codeguru-profiler:GetNotificationConfiguration codeguru-profiler:GetPolicy codeguru-profiler:GetProfile codeguru-profiler:GetRecommendations codeguru-profiler:ListFindingsReports codeguru-profiler:ListProfileTimes codeguru-profiler:ListProfilingGroups codeguru-profiler:PutPermission codeguru-profiler:RemoveNotificationChannel codeguru-profiler:RemovePermission codeguru-profiler:SubmitFeedback codeguru-profiler:UpdateProfilingGroup

服务前缀	操作
codeguru-reviewer	codeguru-reviewer:AssociateRepository codeguru-reviewer:CreateCodeReview codeguru-reviewer:DescribeCodeReview codeguru-reviewer:DescribeRecommendationFeedback codeguru-reviewer:DescribeRepositoryAssociation codeguru-reviewer:DisassociateRepository codeguru-reviewer:ListCodeReviews codeguru-reviewer:ListRecommendationFeedback codeguru-reviewer:ListRecommendations codeguru-reviewer:ListRepositoryAssociations codeguru-reviewer:PutRecommendationFeedback

服务前缀	操作
codepipeline	codepipeline:AcknowledgeJob codepipeline:AcknowledgeThirdPartyJob codepipeline:CreateCustomActionType codepipeline:CreatePipeline codepipeline:DeleteCustomActionType codepipeline:DeletePipeline codepipeline:DeleteWebhook codepipeline:DeregisterWebhookWithThirdParty codepipeline:GetActionType codepipeline:GetJobDetails codepipeline:GetPipeline codepipeline:GetPipelineExecution codepipeline:GetPipelineState codepipeline:GetThirdPartyJobDetails codepipeline:ListActionExecutions codepipeline:ListActionTypes codepipeline:ListPipelineExecutions codepipeline:ListPipelines codepipeline:ListRuleExecutions codepipeline:ListRuleTypes codepipeline:ListWebhooks

服务前缀	操作
	codepipeline:OverrideStageCondition codepipeline:PollForJobs codepipeline:PollForThirdPartyJobs codepipeline:PutActionRevision codepipeline:PutApprovalResult codepipeline:PutJobFailureResult codepipeline:PutJobSuccessResult codepipeline:PutThirdPartyJobFailureResult codepipeline:PutThirdPartyJobSuccessResult codepipeline:PutWebhook codepipeline:RegisterWebhookWithThirdParty codepipeline:RollbackStage codepipeline:StartPipelineExecution codepipeline:StopPipelineExecution codepipeline:UpdateActionType codepipeline:UpdatePipeline

服务前缀	操作
codestar	codestar:AssociateTeamMember
	codestar:CreateProject
	codestar:CreateUserProfile
	codestar:DeleteProject
	codestar:DeleteUserProfile
	codestar:DescribeProject
	codestar:DescribeUserProfile
	codestar:DisassociateTeamMember
	codestar:ListProjects
	codestar:ListResources
	codestar:ListTeamMembers
	codestar:ListUserProfiles
	codestar:UpdateProject
	codestar:UpdateTeamMember
	codestar:UpdateUserProfile

服务前缀	操作
codestar-notifications	codestar-notifications:CreateNotificationRule codestar-notifications:DeleteNotificationRule codestar-notifications:DeleteTarget codestar-notifications:DescribeNotificationRule codestar-notifications:ListEventTypes codestar-notifications:ListNotificationRules codestar-notifications:ListTargets codestar-notifications:Subscribe codestar-notifications:Unsubscribe codestar-notifications:UpdateNotificationRule

服务前缀	操作
cognito-identity	cognito-identity:CreateIdentityPool
	cognito-identity:DeleteIdentities
	cognito-identity:DeleteIdentityPool
	cognito-identity:DescribeIdentity
	cognito-identity:DescribeIdentityPool
	cognito-identity:GetIdentityPoolRoles
	cognito-identity:ListIdentities
	cognito-identity:ListIdentityPools
	cognito-identity:LookupDeveloperIdentity
	cognito-identity:MergeDeveloperIdentities
	cognito-identity:SetIdentityPoolRoles
	cognito-identity:UnlinkDeveloperIdentity
	cognito-identity:UpdateIdentityPool

服务前缀	操作
cognito-idp	cognito-idp:AddCustomAttributes cognito-idp:AdminAddUserToGroup cognito-idp:AdminConfirmSignUp cognito-idp:AdminCreateUser cognito-idp:AdminDeleteUser cognito-idp:AdminDeleteUserAttributes cognito-idp:AdminDisableProviderForUser cognito-idp:AdminDisableUser cognito-idp:AdminEnableUser cognito-idp:AdminForgetDevice cognito-idp:AdminGetDevice cognito-idp:AdminGetUser cognito-idp:AdminInitiateAuth cognito-idp:AdminLinkProviderForUser cognito-idp:AdminListDevices cognito-idp:AdminListGroupsForUser cognito-idp:AdminListUserAuthEvents cognito-idp:AdminRemoveUserFromGroup cognito-idp:AdminResetUserPassword cognito-idp:AdminRespondToAuthChallenge cognito-idp:AdminSetUserMFAPreference

服务前缀	操作
	cognito-idp:AdminSetUserPassword cognito-idp:AdminSetUserSettings cognito-idp:AdminUpdateAuthEventFeedback cognito-idp:AdminUpdateDeviceStatus cognito-idp:AdminUpdateUserAttributes cognito-idp:AdminUserGlobalSignOut cognito-idp:AssociateSoftwareToken cognito-idp:ChangePassword cognito-idp:ConfirmDevice cognito-idp:ConfirmForgotPassword cognito-idp:ConfirmSignUp cognito-idp:CreateGroup cognito-idp:CreateIdentityProvider cognito-idp:CreateManagedLoginBranding cognito-idp:CreateResourceServer cognito-idp:CreateTerms cognito-idp:CreateUserImportJob cognito-idp:CreateUserPool cognito-idp:CreateUserPoolClient cognito-idp:CreateUserPoolDomain cognito-idp:DeleteGroup

服务前缀	操作
	cognito-idp:DeleteIdentityProvider cognito-idp:DeleteManagedLoginBranding cognito-idp:DeleteResourceServer cognito-idp:DeleteTerms cognito-idp:DeleteUser cognito-idp:DeleteUserAttributes cognito-idp:DeleteUserPool cognito-idp:DeleteUserPoolClient cognito-idp:DeleteUserPoolDomain cognito-idp:DescribeIdentityProvider cognito-idp:DescribeManagedLoginBranding cognito-idp:DescribeManagedLoginBrandingByClient cognito-idp:DescribeResourceServer cognito-idp:DescribeRiskConfiguration cognito-idp:DescribeTerms cognito-idp:DescribeUserImportJob cognito-idp:DescribeUserPool cognito-idp:DescribeUserPoolClient cognito-idp:DescribeUserPoolDomain cognito-idp:ForgetDevice cognito-idp:ForgotPassword

服务前缀	操作
	cognito-idp:GetCSVHeader cognito-idp:GetDevice cognito-idp:GetGroup cognito-idp:GetIdentityProviderByIdentifier cognito-idp:GetLogDeliveryConfiguration cognito-idp:GetSigningCertificate cognito-idp:GetUICustomization cognito-idp:GetUser cognito-idp:GetUserAttributeVerificationCode cognito-idp:GetUserPoolMfaConfig cognito-idp:GlobalSignOut cognito-idp:InitiateAuth cognito-idp:ListDevices cognito-idp:ListGroups cognito-idp:ListIdentityProviders cognito-idp:ListResourceServers cognito-idp:ListTerms cognito-idp:ListUserImportJobs cognito-idp:ListUserPoolClients cognito-idp:ListUserPools cognito-idp:ListUsers

服务前缀	操作
	cognito-idp:ListUsersInGroup cognito-idp:ResendConfirmationCode cognito-idp:RespondToAuthChallenge cognito-idp:RevokeToken cognito-idp:SetLogDeliveryConfiguration cognito-idp:SetRiskConfiguration cognito-idp:SetUICustomization cognito-idp:SetUserMFAPreference cognito-idp:SetUserPoolMfaConfig cognito-idp:SetUserSettings cognito-idp:SignUp cognito-idp:StartUserImportJob cognito-idp:StopUserImportJob cognito-idp:UpdateAuthEventFeedback cognito-idp:UpdateDeviceStatus cognito-idp:UpdateGroup cognito-idp:UpdateIdentityProvider cognito-idp:UpdateResourceServer cognito-idp:UpdateTerms cognito-idp:UpdateUserAttributes cognito-idp:UpdateUserPool

服务前缀	操作
	cognito-idp:UpdateUserPoolClient cognito-idp:UpdateUserPoolDomain cognito-idp:VerifySoftwareToken cognito-idp:VerifyUserAttribute
cognito-sync	cognito-sync:BulkPublish cognito-sync:DeleteDataset cognito-sync:DescribeDataset cognito-sync:DescribeIdentityPoolUsage cognito-sync:DescribeIdentityUsage cognito-sync:GetBulkPublishDetails cognito-sync:GetCognitoEvents cognito-sync:GetIdentityPoolConfiguration cognito-sync:ListDatasets cognito-sync:ListIdentityPoolUsage cognito-sync:ListRecords cognito-sync:RegisterDevice cognito-sync:SetCognitoEvents cognito-sync:SetIdentityPoolConfiguration cognito-sync:SubscribeToDataset cognito-sync:UnsubscribeFromDataset cognito-sync:UpdateRecords

服务前缀	操作
comprehendmedical	comprehendmedical:DescribeEntitiesDetectionV2Job comprehendmedical:DescribeICD10CMInferenceJob comprehendmedical:DescribePHIDetectionJob comprehendmedical:DescribeRxNormInferenceJob comprehendmedical:DescribeSNOMEDCTInferenceJob comprehendmedical:DetectEntitiesV2 comprehendmedical:DetectPHI comprehendmedical:InferICD10CM comprehendmedical:InferRxNorm comprehendmedical:InferSNOMEDCT comprehendmedical:ListEntitiesDetectionV2Jobs comprehendmedical:ListICD10CMInferenceJobs comprehendmedical:ListPHIDetectionJobs comprehendmedical:ListRxNormInferenceJobs comprehendmedical:ListSNOMEDCTInferenceJobs comprehendmedical:StartEntitiesDetectionV2Job comprehendmedical:StartICD10CMInferenceJob comprehendmedical:StartPHIDetectionJob comprehendmedical:StartRxNormInferenceJob comprehendmedical:StartSNOMEDCTInferenceJob comprehendmedical:StopEntitiesDetectionV2Job

服务前缀	操作
	comprehendmedical:StopICD10CMInferenceJob comprehendmedical:StopPHIDetectionJob comprehendmedical:StopRxNormInferenceJob comprehendmedical:StopSNOMEDCTInferenceJob

服务前缀	操作
compute-optimizer	compute-optimizer:DeleteRecommendationPreferences
	compute-optimizer:DescribeRecommendationExportJobs
	compute-optimizer:ExportAutoScalingGroupRecommendations
	compute-optimizer:ExportEBSVolumeRecommendations
	compute-optimizer:ExportEC2InstanceRecommendations
	compute-optimizer:ExportECSServiceRecommendations
	compute-optimizer:ExportIdleRecommendations
	compute-optimizer:ExportLambdaFunctionRecommendations
	compute-optimizer:ExportLicenseRecommendations
	compute-optimizer:ExportRDSDatabaseRecommendations
	compute-optimizer:GetEC2RecommendationProjectedMetrics
	compute-optimizer:GetECSServiceRecommendationProjectedMetrics
	compute-optimizer:GetEffectiveRecommendationPreferences
	compute-optimizer:GetEnrollmentStatus
	compute-optimizer:GetEnrollmentStatusesForOrganization
	compute-optimizer:GetRDSDatabaseRecommendationProjectedMetrics
	compute-optimizer:GetRecommendationPreferences
	compute-optimizer:GetRecommendationSummaries
	compute-optimizer:PutRecommendationPreferences

服务前缀	操作
	compute-optimizer:UpdateEnrollmentStatus

服务前缀	操作
config	config:BatchGetResourceConfig config:DeleteAggregationAuthorization config:DeleteConfigRule config:DeleteConfigurationAggregator config:DeleteConfigurationRecorder config:DeleteConformancePack config:DeleteDeliveryChannel config:DeleteEvaluationResults config:DeleteOrganizationConfigRule config:DeleteOrganizationConformancePack config:DeletePendingAggregationRequest config:DeleteRemediationConfiguration config:DeleteRemediationExceptions config:DeleteResourceConfig config:DeleteRetentionConfiguration config:DeleteStoredQuery config:DeliverConfigSnapshot config:DescribeAggregateComplianceByConfigRules config:DescribeAggregateComplianceByConformancePacks config:DescribeAggregationAuthorizations config:DescribeComplianceByConfigRule

服务前缀	操作
	config:DescribeComplianceByResource config:DescribeConfigRuleEvaluationStatus config:DescribeConfigRules config:DescribeConfigurationAggregatorSourcesStatus config:DescribeConfigurationAggregators config:DescribeConfigurationRecorderStatus config:DescribeConfigurationRecorders config:DescribeConformancePackCompliance config:DescribeConformancePackStatus config:DescribeConformancePacks config:DescribeDeliveryChannelStatus config:DescribeDeliveryChannels config:DescribeOrganizationConfigRuleStatuses config:DescribeOrganizationConfigRules config:DescribeOrganizationConformancePackStatuses config:DescribeOrganizationConformancePacks config:DescribePendingAggregationRequests config:DescribeRemediationConfigurations config:DescribeRemediationExceptions config:DescribeRemediationExecutionStatus config:DescribeRetentionConfigurations

服务前缀	操作
	config:GetComplianceDetailsByConfigRule config:GetComplianceDetailsByResource config:GetComplianceSummaryByConfigRule config:GetComplianceSummaryByResourceType config:GetConformancePackComplianceDetails config:GetConformancePackComplianceSummary config:GetCustomRulePolicy config:GetDiscoveredResourceCounts config:GetOrganizationConfigRuleDetailedStatus config:GetOrganizationConformancePackDetailedStatus config:GetOrganizationCustomRulePolicy config:GetResourceConfigHistory config:GetResourceEvaluationSummary config:GetStoredQuery config:ListConfigurationRecorders config:ListConformancePackComplianceScores config:ListDiscoveredResources config:ListResourceEvaluations config:ListStoredQueries config:PutConfigRule config:PutConfigurationAggregator

服务前缀	操作
	config:PutConfigurationRecorder config:PutConformancePack config:PutDeliveryChannel config:PutEvaluations config:PutExternalEvaluation config:PutOrganizationConfigRule config:PutOrganizationConformancePack config:PutRemediationConfigurations config:PutRemediationExceptions config:PutResourceConfig config:PutRetentionConfiguration config:PutStoredQuery config:SelectResourceConfig config:StartConfigRulesEvaluation config:StartConfigurationRecorder config:StartRemediationExecution config:StartResourceEvaluation config:StopConfigurationRecorder

服务前缀	操作
connect	connect:ActivateEvaluationForm
	connect:AssociateAnalyticsDataSet
	connect:AssociateApprovedOrigin
	connect:AssociateBot
	connect:AssociateContactWithUser
	connect:AssociateDefaultVocabulary
	connect:AssociateEmailAddressAlias
	connect:AssociateFlow
	connect:AssociateInstanceStorageConfig
	connect:AssociateLambdaFunction
	connect:AssociateLexBot
	connect:AssociatePhoneNumberContactFlow
	connect:AssociateQueueQuickConnects
	connect:AssociateRoutingProfileQueues
	connect:AssociateSecurityKey
	connect:AssociateUserProficiencies
	connect:BatchAssociateAnalyticsDataSet
	connect:BatchCreateDataTableValue
	connect:BatchDeleteDataTableValue
	connect:BatchDescribeDataTableValue
	connect:BatchDisassociateAnalyticsDataSet

服务前缀	操作
	connect:BatchGetFlowAssociation
	connect:BatchPutContact
	connect:BatchUpdateDataTableValue
	connect:ClaimPhoneNumber
	connect:CreateAgentStatus
	connect:CreateContact
	connect:CreateContactFlow
	connect:CreateContactFlowModule
	connect:CreateContactFlowModuleAlias
	connect:CreateContactFlowModuleVersion
	connect:CreateContactFlowVersion
	connect:CreateDataTable
	connect:CreateDataTableAttribute
	connect:CreateEmailAddress
	connect:CreateEvaluationForm
	connect:CreateHoursOfOperation
	connect:CreateInstance
	connect:CreateIntegrationAssociation
	connect:CreateParticipant
	connect:CreatePersistentContactAssociation
	connect:CreatePredefinedAttribute

服务前缀	操作
	connect:CreatePrompt
	connect:CreatePushNotificationRegistration
	connect:CreateQueue
	connect:CreateQuickConnect
	connect:CreateRoutingProfile
	connect:CreateRule
	connect:CreateSecurityProfile
	connect:CreateTaskTemplate
	connect:CreateTrafficDistributionGroup
	connect:CreateUseCase
	connect:CreateUser
	connect:CreateUserHierarchyGroup
	connect:CreateView
	connect:CreateViewVersion
	connect:CreateVocabulary
	connect:CreateWorkspace
	connect:DeactivateEvaluationForm
	connect:DeleteContactEvaluation
	connect:DeleteContactFlow
	connect:DeleteContactFlowModule
	connect:DeleteContactFlowModuleAlias

服务前缀	操作
	connect:DeleteContactFlowModuleVersion
	connect:DeleteContactFlowVersion
	connect:DeleteDataTable
	connect:DeleteDataTableAttribute
	connect:DeleteEmailAddress
	connect:DeleteEvaluationForm
	connect:DeleteHoursOfOperation
	connect:DeleteHoursOfOperationOverride
	connect:DeleteInstance
	connect:DeleteIntegrationAssociation
	connect:DeletePredefinedAttribute
	connect:DeletePrompt
	connect:DeletePushNotificationRegistration
	connect:DeleteQueue
	connect:DeleteQuickConnect
	connect:DeleteRoutingProfile
	connect:DeleteRule
	connect:DeleteSecurityProfile
	connect:DeleteTaskTemplate
	connect:DeleteTrafficDistributionGroup
	connect:DeleteUseCase

服务前缀	操作
	connect:DeleteUser
	connect:DeleteUserHierarchyGroup
	connect:DeleteView
	connect:DeleteVocabulary
	connect:DeleteWorkspace
	connect:DeleteWorkspaceMedia
	connect:DescribeAuthenticationProfile
	connect:DescribeContactFlowModuleAlias
	connect:DescribeDataTableAttribute
	connect:DescribeHoursOfOperationOverride
	connect:DescribeInstanceAttribute
	connect:DescribeInstanceStorageConfig
	connect:DescribePhoneNumber
	connect:DescribeRule
	connect:DescribeTrafficDistributionGroup
	connect:DescribeUserHierarchyStructure
	connect:DescribeVocabulary
	connect:DisassociateAnalyticsDataSet
	connect:DisassociateApprovedOrigin
	connect:DisassociateBot
	connect:DisassociateEmailAddressAlias

服务前缀	操作
	connect:DisassociateFlow
	connect:DisassociateInstanceStorageConfig
	connect:DisassociateLambdaFunction
	connect:DisassociateLexBot
	connect:DisassociatePhoneNumberContactFlow
	connect:DisassociateQueueQuickConnects
	connect:DisassociateRoutingProfileQueues
	connect:DisassociateSecurityKey
	connect:DisassociateUserProficiencies
	connect:DismissUserContact
	connect:EvaluateDataTableValues
	connect:GetContactAttributes
	connect:GetContactMetrics
	connect:GetCurrentMetricData
	connect:GetCurrentUserData
	connect:GetEffectiveHoursOfOperations
	connect:GetFederationToken
	connect:GetFlowAssociation
	connect:GetMetricData
	connect:GetMetricDataV2
	connect:GetPromptFile

服务前缀	操作
	connect:GetTaskTemplate
	connect:GetTrafficDistribution
	connect:ImportPhoneNumber
	connect:ImportWorkspaceMedia
	connect:ListAnalyticsDataAssociations
	connect:ListAnalyticsDataLakeDataSets
	connect:ListApprovedOrigins
	connect:ListAssociatedContacts
	connect:ListAuthenticationProfiles
	connect:ListBots
	connect:ListContactEvaluations
	connect:ListContactFlowModuleAliases
	connect:ListContactFlowModuleVersions
	connect:ListContactFlowModules
	connect:ListContactFlowVersions
	connect:ListContactFlows
	connect:ListContactReferences
	connect:ListDataTableAttributes
	connect:ListDataTablePrimaryValues
	connect:ListDataTableValues
	connect:ListDataTables

服务前缀	操作
	connect:ListDefaultVocabularies
	connect:ListEvaluationFormVersions
	connect:ListEvaluationForms
	connect:ListFlowAssociations
	connect:ListHoursOfOperations
	connect:ListInstanceAttributes
	connect:ListInstanceStorageConfigs
	connect:ListIntegrationAssociations
	connect:ListLambdaFunctions
	connect:ListLexBots
	connect:ListPhoneNumbers
	connect:ListPhoneNumbersV2
	connect:ListPredefinedAttributes
	connect:ListPrompts
	connect:ListQueueQuickConnects
	connect:ListQueues
	connect:ListQuickConnects
	connect:ListRealtimeContactAnalysisSegmentsV2
	connect:ListRoutingProfileManualAssignmentQueues
	connect:ListRoutingProfileQueues
	connect:ListRoutingProfiles

服务前缀	操作
	connect:ListRules
	connect:ListSecurityKeys
	connect:ListSecurityProfileApplications
	connect:ListSecurityProfileFlowModules
	connect:ListSecurityProfilePermissions
	connect:ListSecurityProfiles
	connect:ListTaskTemplates
	connect:ListTrafficDistributionGroups
	connect:ListUseCases
	connect:ListUserHierarchyGroups
	connect:ListUsers
	connect:ListViewVersions
	connect:ListViews
	connect:ListWorkspaceMedia
	connect:ListWorkspacePages
	connect:ListWorkspaces
	connect:MonitorContact
	connect:PauseContact
	connect:PutUserStatus
	connect:ReleasePhoneNumber
	connect:ReplicateInstance

服务前缀	操作
	connect:ResumeContact
	connect:ResumeContactRecording
	connect:SearchAgentStatuses
	connect:SearchAvailablePhoneNumbers
	connect:SearchContactEvaluations
	connect:SearchContactFlowModules
	connect:SearchContactFlows
	connect:SearchContacts
	connect:SearchDataTables
	connect:SearchEmailAddresses
	connect:SearchEvaluationForms
	connect:SearchHoursOfOperations
	connect:SearchPredefinedAttributes
	connect:SearchPrompts
	connect:SearchQueues
	connect:SearchQuickConnects
	connect:SearchRoutingProfiles
	connect:SearchSecurityProfiles
	connect:SearchUserHierarchyGroups
	connect:SearchViews
	connect:SearchVocabularies

服务前缀	操作
	connect:SearchWorkspaceAssociations
	connect:SearchWorkspaces
	connect:SendChatIntegrationEvent
	connect:SendOutboundEmail
	connect:StartChatContact
	connect:StartContactEvaluation
	connect:StartContactMediaProcessing
	connect:StartContactRecording
	connect:StartContactStreaming
	connect:StartEmailContact
	connect:StartOutboundChatContact
	connect:StartOutboundEmailContact
	connect:StartOutboundVoiceContact
	connect:StartScreenSharing
	connect:StartTaskContact
	connect:StartWebRTCContact
	connect:StopContact
	connect:StopContactMediaProcessing
	connect:StopContactRecording
	connect:StopContactStreaming
	connect:SubmitContactEvaluation

服务前缀	操作
	connect:SuspendContactRecording
	connect:TransferContact
	connect:UpdateAgentStatus
	connect:UpdateAuthenticationProfile
	connect:UpdateContact
	connect:UpdateContactAttributes
	connect:UpdateContactEvaluation
	connect:UpdateContactFlowContent
	connect:UpdateContactFlowMetadata
	connect:UpdateContactFlowModuleAlias
	connect:UpdateContactFlowModuleContent
	connect:UpdateContactFlowModuleMetadata
	connect:UpdateContactFlowName
	connect:UpdateContactRoutingData
	connect:UpdateContactSchedule
	connect:UpdateDataTableAttribute
	connect:UpdateDataTableMetadata
	connect:UpdateDataTablePrimaryValues
	connect:UpdateEmailAddressMetadata
	connect:UpdateEvaluationForm
	connect:UpdateHoursOfOperation

服务前缀	操作
	connect:UpdateHoursOfOperationOverride
	connect:UpdateInstanceAttribute
	connect:UpdateInstanceStorageConfig
	connect:UpdateParticipantAuthentication
	connect:UpdateParticipantRoleConfig
	connect:UpdatePhoneNumber
	connect:UpdatePhoneNumberMetadata
	connect:UpdatePredefinedAttribute
	connect:UpdatePrompt
	connect:UpdateQueueHoursOfOperation
	connect:UpdateQueueMaxContacts
	connect:UpdateQueueName
	connect:UpdateQueueOutboundCallerConfig
	connect:UpdateQueueOutboundEmailConfig
	connect:UpdateQueueStatus
	connect:UpdateQuickConnectConfig
	connect:UpdateQuickConnectName
	connect:UpdateRoutingProfileAgentAvailabilityTimer
	connect:UpdateRoutingProfileConcurrency
	connect:UpdateRoutingProfileDefaultOutboundQueue
	connect:UpdateRoutingProfileName

服务前缀	操作
	connect:UpdateRoutingProfileQueues
	connect:UpdateRule
	connect:UpdateSecurityProfile
	connect:UpdateTaskTemplate
	connect:UpdateTrafficDistribution
	connect:UpdateUserHierarchy
	connect:UpdateUserHierarchyGroupName
	connect:UpdateUserHierarchyStructure
	connect:UpdateUserIdentityInfo
	connect:UpdateUserPhoneConfig
	connect:UpdateUserProficiencies
	connect:UpdateUserRoutingProfile
	connect:UpdateUserSecurityProfiles
	connect:UpdateViewContent
	connect:UpdateViewMetadata
	connect:UpdateWorkspaceMetadata
	connect:UpdateWorkspaceTheme
	connect:UpdateWorkspaceVisibility

服务前缀	操作
cur	cur:DeleteReportDefinition cur:DescribeReportDefinitions cur:ModifyReportDefinition cur:PutReportDefinition

服务前缀	操作
databrew	databrew:BatchDeleteRecipeVersion
	databrew:CreateDataset
	databrew:CreateProfileJob
	databrew:CreateProject
	databrew:CreateRecipe
	databrew:CreateRecipeJob
	databrew:CreateRuleset
	databrew:CreateSchedule
	databrew:DeleteDataset
	databrew:DeleteJob
	databrew:DeleteProject
	databrew:DeleteRecipeVersion
	databrew:DeleteRuleset
	databrew:DeleteSchedule
	databrew:DescribeDataset
	databrew:DescribeJob
	databrew:DescribeJobRun
	databrew:DescribeProject
	databrew:DescribeRecipe
	databrew:DescribeRuleset
	databrew:DescribeSchedule

服务前缀	操作
	databrew:ListDatasets
	databrew:ListJobRuns
	databrew:ListJobs
	databrew:ListProjects
	databrew:ListRecipeVersions
	databrew:ListRecipes
	databrew:ListRulesets
	databrew:ListSchedules
	databrew:PublishRecipe
	databrew:SendProjectSessionAction
	databrew:StartJobRun
	databrew:StartProjectSession
	databrew:StopJobRun
	databrew:UpdateDataset
	databrew:UpdateProfileJob
	databrew:UpdateProject
	databrew:UpdateRecipe
	databrew:UpdateRecipeJob
	databrew:UpdateRuleset
	databrew:UpdateSchedule

服务前缀	操作
dataexchange	dataexchange:AcceptDataGrant dataexchange:CancelJob dataexchange:CreateDataGrant dataexchange:CreateDataSet dataexchange:CreateEventAction dataexchange:CreateJob dataexchange:CreateRevision dataexchange:DeleteAsset dataexchange:DeleteDataGrant dataexchange:DeleteEventAction dataexchange:DeleteRevision dataexchange:GetDataGrant dataexchange:GetEventAction dataexchange:GetJob dataexchange:GetReceivedDataGrant dataexchange:ListDataGrants dataexchange:ListDataSetRevisions dataexchange:ListDataSets dataexchange:ListEventActions dataexchange:ListJobs dataexchange:ListReceivedDataGrants

服务前缀	操作
	dataexchange:ListRevisionAssets dataexchange:RevokeRevision dataexchange:SendDataSetNotification dataexchange:StartJob dataexchange:UpdateAsset dataexchange:UpdateDataSet dataexchange:UpdateEventAction dataexchange:UpdateRevision

服务前缀	操作
datapipeline	datapipeline:ActivatePipeline datapipeline:CreatePipeline datapipeline:DeactivatePipeline datapipeline:DeletePipeline datapipeline:DescribeObjects datapipeline:DescribePipelines datapipeline:EvaluateExpression datapipeline:GetPipelineDefinition datapipeline:ListPipelines datapipeline:PollForTask datapipeline:PutPipelineDefinition datapipeline:QueryObjects datapipeline:ReportTaskProgress datapipeline:ReportTaskRunnerHeartbeat datapipeline:SetStatus datapipeline:SetTaskStatus datapipeline:ValidatePipelineDefinition

服务前缀	操作
dax	dax:CreateCluster dax:DecreaseReplicationFactor dax:DeleteCluster dax:DeleteParameterGroup dax:DeleteSubnetGroup dax:DescribeClusters dax:DescribeDefaultParameters dax:DescribeEvents dax:DescribeParameterGroups dax:DescribeParameters dax:DescribeSubnetGroups dax:IncreaseReplicationFactor dax:RebootNode dax:UpdateCluster dax:UpdateParameterGroup dax:UpdateSubnetGroup

服务前缀	操作
devicefarm	devicefarm:CreateDevicePool
	devicefarm:CreateInstanceProfile
	devicefarm:CreateNetworkProfile
	devicefarm:CreateProject
	devicefarm:CreateRemoteAccessSession
	devicefarm:CreateTestGridProject
	devicefarm:CreateTestGridUrl
	devicefarm:CreateUpload
	devicefarm:CreateVPCEConfiguration
	devicefarm:DeleteDevicePool
	devicefarm:DeleteInstanceProfile
	devicefarm:DeleteNetworkProfile
	devicefarm:DeleteProject
	devicefarm:DeleteRemoteAccessSession
	devicefarm:DeleteRun
	devicefarm:DeleteTestGridProject
	devicefarm:DeleteUpload
	devicefarm:DeleteVPCEConfiguration
	devicefarm:GetAccountSettings
	devicefarm:GetDevice
	devicefarm:GetDeviceInstance

服务前缀	操作
	devicefarm:GetDevicePool
	devicefarm:GetDevicePoolCompatibility
	devicefarm:GetInstanceProfile
	devicefarm:GetJob
	devicefarm:GetNetworkProfile
	devicefarm:GetOfferingStatus
	devicefarm:GetProject
	devicefarm:GetRemoteAccessSession
	devicefarm:GetRun
	devicefarm:GetSuite
	devicefarm:GetTest
	devicefarm:GetTestGridProject
	devicefarm:GetTestGridSession
	devicefarm:GetUpload
	devicefarm:GetVPCEConfiguration
	devicefarm:ListArtifacts
	devicefarm:ListDeviceInstances
	devicefarm:ListDevicePools
	devicefarm:ListDevices
	devicefarm:ListInstanceProfiles
	devicefarm:ListJobs

服务前缀	操作
	devicefarm:ListNetworkProfiles
	devicefarm:ListOfferingPromotions
	devicefarm:ListOfferingTransactions
	devicefarm:ListOfferings
	devicefarm:ListProjects
	devicefarm:ListRemoteAccessSessions
	devicefarm:ListRuns
	devicefarm:ListSamples
	devicefarm:ListSuites
	devicefarm:ListTestGridProjects
	devicefarm:ListTestGridSessionActions
	devicefarm:ListTestGridSessionArtifacts
	devicefarm:ListTestGridSessions
	devicefarm:ListTests
	devicefarm:ListUniqueProblems
	devicefarm:ListUploads
	devicefarm:ListVPCEConfigurations
	devicefarm:PurchaseOffering
	devicefarm:RenewOffering
	devicefarm:ScheduleRun
	devicefarm:StopJob

服务前缀	操作
	devicefarm:StopRemoteAccessSession
	devicefarm:StopRun
	devicefarm:UpdateDeviceInstance
	devicefarm:UpdateDevicePool
	devicefarm:UpdateInstanceProfile
	devicefarm:UpdateNetworkProfile
	devicefarm:UpdateProject
	devicefarm:UpdateTestGridProject
	devicefarm:UpdateUpload
	devicefarm:UpdateVPCEConfiguration

服务前缀	操作
devops-guru	devops-guru:AddNotificationChannel devops-guru:DeleteInsight devops-guru:DescribeAccountHealth devops-guru:DescribeAccountOverview devops-guru:DescribeAnomaly devops-guru:DescribeEventSourcesConfig devops-guru:DescribeFeedback devops-guru:DescribeInsight devops-guru:DescribeOrganizationHealth devops-guru:DescribeOrganizationOverview devops-guru:DescribeOrganizationResourceCollectionHealth devops-guru:DescribeResourceCollectionHealth devops-guru:DescribeServiceIntegration devops-guru:GetCostEstimation devops-guru:GetResourceCollection devops-guru:ListAnomaliesForInsight devops-guru:ListAnomalousLogGroups devops-guru:ListEvents devops-guru:ListInsights devops-guru:ListMonitoredResources devops-guru:ListNotificationChannels

服务前缀	操作
	devops-guru:ListOrganizationInsights devops-guru:ListRecommendations devops-guru:PutFeedback devops-guru:RemoveNotificationChannel devops-guru:SearchInsights devops-guru:SearchOrganizationInsights devops-guru:StartCostEstimation devops-guru:UpdateEventSourcesConfig devops-guru:UpdateResourceCollection devops-guru:UpdateServiceIntegration

服务前缀	操作
directconnect	directconnect:AcceptDirectConnectGatewayAssociationProposal
	directconnect:AllocateConnectionOnInterconnect
	directconnect:AllocateHostedConnection
	directconnect:AllocatePrivateVirtualInterface
	directconnect:AllocatePublicVirtualInterface
	directconnect:AllocateTransitVirtualInterface
	directconnect:AssociateConnectionWithLag
	directconnect:AssociateHostedConnection
	directconnect:AssociateMacSecKey
	directconnect:AssociateVirtualInterface
	directconnect:ConfirmConnection
	directconnect:ConfirmCustomerAgreement
	directconnect:ConfirmPrivateVirtualInterface
	directconnect:ConfirmPublicVirtualInterface
	directconnect:ConfirmTransitVirtualInterface
	directconnect:CreateBGPPeer
	directconnect:CreateConnection
	directconnect:CreateDirectConnectGateway
	directconnect:CreateDirectConnectGatewayAssociation
	directconnect:CreateDirectConnectGatewayAssociationProposal
	directconnect:CreateInterconnect

服务前缀	操作
	directconnect:CreateLag
	directconnect:CreatePrivateVirtualInterface
	directconnect:CreatePublicVirtualInterface
	directconnect:CreateTransitVirtualInterface
	directconnect:DeleteBGPPeer
	directconnect:DeleteConnection
	directconnect:DeleteDirectConnectGateway
	directconnect:DeleteDirectConnectGatewayAssociation
	directconnect:DeleteDirectConnectGatewayAssociationProposal
	directconnect:DeleteInterconnect
	directconnect:DeleteLag
	directconnect:DeleteVirtualInterface
	directconnect:DescribeConnectionLoa
	directconnect:DescribeConnections
	directconnect:DescribeConnectionsOnInterconnect
	directconnect:DescribeCustomerMetadata
	directconnect:DescribeDirectConnectGatewayAssociationProposals
	directconnect:DescribeDirectConnectGatewayAssociations
	directconnect:DescribeDirectConnectGatewayAttachments
	directconnect:DescribeDirectConnectGateways
	directconnect:DescribeHostedConnections

服务前缀	操作
	directconnect:DescribeInterconnectLoa
	directconnect:DescribeInterconnects
	directconnect:DescribeLags
	directconnect:DescribeLoa
	directconnect:DescribeLocations
	directconnect:DescribeRouterConfiguration
	directconnect:DescribeVirtualGateways
	directconnect:DescribeVirtualInterfaces
	directconnect:DisassociateConnectionFromLag
	directconnect:DisassociateMacSecKey
	directconnect:ListVirtualInterfaceTestHistory
	directconnect:StartBgpFailoverTest
	directconnect:StopBgpFailoverTest
	directconnect:UpdateConnection
	directconnect:UpdateDirectConnectGateway
	directconnect:UpdateDirectConnectGatewayAssociation
	directconnect:UpdateLag
	directconnect:UpdateVirtualInterfaceAttributes

服务前缀	操作
dlm	dlm:CreateLifecyclePolicy dlm:DeleteLifecyclePolicy dlm:GetLifecyclePolicies dlm:GetLifecyclePolicy dlm:UpdateLifecyclePolicy

服务前缀	操作
dms	dms:ApplyPendingMaintenanceAction
	dms:AssociateExtensionPack
	dms:BatchStartRecommendations
	dms:CancelMetadataModelCreation
	dms:CancelReplicationTaskAssessmentRun
	dms:CreateDataProvider
	dms:CreateEndpoint
	dms:CreateEventSubscription
	dms:CreateInstanceProfile
	dms:CreateMigrationProject
	dms:CreateReplicationConfig
	dms:CreateReplicationInstance
	dms:CreateReplicationSubnetGroup
	dms:CreateReplicationTask
	dms:DeleteCertificate
	dms:DeleteConnection
	dms:DeleteDataMigration
	dms:DeleteDataProvider
	dms:DeleteEndpoint
	dms:DeleteEventSubscription
	dms:DeleteFleetAdvisorCollector

服务前缀	操作
	dms:DeleteFleetAdvisorDatabases
	dms:DeleteInstanceProfile
	dms:DeleteMigrationProject
	dms:DeleteReplicationConfig
	dms:DeleteReplicationInstance
	dms:DeleteReplicationSubnetGroup
	dms:DeleteReplicationTask
	dms:DeleteReplicationTaskAssessmentRun
	dms:DescribeAccountAttributes
	dms:DescribeApplicableIndividualAssessments
	dms:DescribeCertificates
	dms:DescribeConnections
	dms:DescribeDataMigrations
	dms:DescribeEndpointSettings
	dms:DescribeEndpointTypes
	dms:DescribeEndpoints
	dms:DescribeEngineVersions
	dms:DescribeEventCategories
	dms:DescribeEventSubscriptions
	dms:DescribeEvents
	dms:DescribeFleetAdvisorCollectors

服务前缀	操作
	dms:DescribeFleetAdvisorDatabases
	dms:DescribeFleetAdvisorLsaAnalysis
	dms:DescribeFleetAdvisorSchemaObjectSummary
	dms:DescribeFleetAdvisorSchemas
	dms:DescribeMetadataModel
	dms:DescribeMetadataModelChildren
	dms:DescribeMetadataModelCreations
	dms:DescribeMetadataModelImports
	dms:DescribeOrderableReplicationInstances
	dms:DescribePendingMaintenanceActions
	dms:DescribeRecommendationLimitations
	dms:DescribeRecommendations
	dms:DescribeRefreshSchemasStatus
	dms:DescribeReplicationConfigs
	dms:DescribeReplicationInstanceTaskLogs
	dms:DescribeReplicationInstances
	dms:DescribeReplicationSubnetGroups
	dms:DescribeReplicationTableStatistics
	dms:DescribeReplicationTaskAssessmentResults
	dms:DescribeReplicationTaskAssessmentRuns
	dms:DescribeReplicationTaskIndividualAssessments

服务前缀	操作
	dms:DescribeReplicationTasks
	dms:DescribeReplications
	dms:DescribeSchemas
	dms:DescribeTableStatistics
	dms:ExportMetadataModelAssessment
	dms:ImportCertificate
	dms:ListDataProviders
	dms:ListExtensionPacks
	dms:ListInstanceProfiles
	dms:ListMetadataModelAssessments
	dms:ListMetadataModelConversions
	dms:ListMetadataModelExports
	dms:ListMigrationProjects
	dms:ModifyDataMigration
	dms:ModifyEndpoint
	dms:ModifyEventSubscription
	dms:ModifyReplicationConfig
	dms:ModifyReplicationInstance
	dms:ModifyReplicationSubnetGroup
	dms:ModifyReplicationTask
	dms:MoveReplicationTask

服务前缀	操作
	dms:RebootReplicationInstance
	dms:RefreshSchemas
	dms:ReloadReplicationTables
	dms:ReloadTables
	dms:RunFleetAdvisorLsaAnalysis
	dms:StartMetadataModelAssessment
	dms:StartMetadataModelConversion
	dms:StartMetadataModelCreation
	dms:StartMetadataModelExportAsScripts
	dms:StartMetadataModelExportToTarget
	dms:StartRecommendations
	dms:StartReplication
	dms:StartReplicationTask
	dms:StartReplicationTaskAssessment
	dms:StopDataMigration
	dms:StopReplicationTask
	dms:TestConnection
	dms:UpdateConversionConfiguration
	dms:UpdateDataProvider
	dms:UpdateInstanceProfile
	dms:UpdateMigrationProject

服务前缀	操作
	dms:UpdateSubscriptionsToEventBridge
docdb-elastic	docdb-elastic:ApplyPendingMaintenanceAction docdb-elastic:CopyClusterSnapshot docdb-elastic:DeleteCluster docdb-elastic:DeleteClusterSnapshot docdb-elastic:GetCluster docdb-elastic:GetClusterSnapshot docdb-elastic:GetPendingMaintenanceAction docdb-elastic:ListClusterSnapshots docdb-elastic:ListClusters docdb-elastic:ListPendingMaintenanceActions docdb-elastic:RestoreClusterFromSnapshot docdb-elastic:StartCluster docdb-elastic:StopCluster docdb-elastic:UpdateCluster

服务前缀	操作
dynamodb	dynamodb:CreateBackup
	dynamodb:CreateGlobalTable
	dynamodb:CreateTable
	dynamodb:DeleteBackup
	dynamodb:DeleteTable
	dynamodb:DescribeBackup
	dynamodb:DescribeContinuousBackups
	dynamodb:DescribeContributorInsights
	dynamodb:DescribeEndpoints
	dynamodb:DescribeExport
	dynamodb:DescribeGlobalTable
	dynamodb:DescribeGlobalTableSettings
	dynamodb:DescribeImport
	dynamodb:DescribeKinesisStreamingDestination
	dynamodb:DescribeLimits
	dynamodb:DescribeStream
	dynamodb:DescribeTable
	dynamodb:DescribeTableReplicaAutoScaling
	dynamodb:DescribeTimeToLive
	dynamodb:DisableKinesisStreamingDestination
	dynamodb:EnableKinesisStreamingDestination

服务前缀	操作
	dynamodb:ExportTableToPointInTime
	dynamodb:GetResourcePolicy
	dynamodb:ImportTable
	dynamodb:ListBackups
	dynamodb:ListContributorInsights
	dynamodb:ListExports
	dynamodb:ListGlobalTables
	dynamodb:ListImports
	dynamodb:ListStreams
	dynamodb:ListTables
	dynamodb:RestoreTableFromBackup
	dynamodb:RestoreTableToPointInTime
	dynamodb:UpdateContinuousBackups
	dynamodb:UpdateContributorInsights
	dynamodb:UpdateGlobalTable
	dynamodb:UpdateGlobalTableSettings
	dynamodb:UpdateKinesisStreamingDestination
	dynamodb:UpdateTable
	dynamodb:UpdateTableReplicaAutoScaling
	dynamodb:UpdateTimeToLive

服务前缀	操作
ebs	ebs:CompleteSnapshot ebs:StartSnapshot

服务前缀	操作
ec2	ec2:AcceptAddressTransfer
	ec2:AcceptCapacityReservationBillingOwnership
	ec2:AcceptReservedInstancesExchangeQuote
	ec2:AcceptTransitGatewayMulticastDomainAssociations
	ec2:AcceptTransitGatewayPeeringAttachment
	ec2:AcceptTransitGatewayVpcAttachment
	ec2:AcceptVpcEndpointConnections
	ec2:AcceptVpcPeeringConnection
	ec2:AdvertiseByoipCidr
	ec2:AllocateAddress
	ec2:AllocateHosts
	ec2:AllocateIpamPoolCidr
	ec2:ApplySecurityGroupsToClientVpnTargetNetwork
	ec2:AssignIpv6Addresses
	ec2:AssignPrivateIpAddresses
	ec2:AssignPrivateNatGatewayAddress
	ec2:AssociateAddress
	ec2:AssociateCapacityReservationBillingOwner
	ec2:AssociateClientVpnTargetNetwork
	ec2:AssociateDhcpOptions
	ec2:AssociateEnclaveCertificateIamRole

服务前缀	操作
	ec2:AssociateIamInstanceProfile
	ec2:AssociateInstanceEventWindow
	ec2:AssociateIpamByoasn
	ec2:AssociateIpamResourceDiscovery
	ec2:AssociateNatGatewayAddress
	ec2:AssociateRouteServer
	ec2:AssociateRouteTable
	ec2:AssociateSecurityGroupVpc
	ec2:AssociateSubnetCidrBlock
	ec2:AssociateTransitGatewayMulticastDomain
	ec2:AssociateTransitGatewayPolicyTable
	ec2:AssociateTransitGatewayRouteTable
	ec2:AssociateTrunkInterface
	ec2:AssociateVpcCidrBlock
	ec2:AttachClassicLinkVpc
	ec2:AttachInternetGateway
	ec2:AttachNetworkInterface
	ec2:AttachVerifiedAccessTrustProvider
	ec2:AttachVolume
	ec2:AttachVpnGateway
	ec2:AuthorizeClientVpnIngress

服务前缀	操作
	ec2:AuthorizeSecurityGroupEgress
	ec2:AuthorizeSecurityGroupIngress
	ec2:BundleInstance
	ec2:CancelBundleTask
	ec2:CancelCapacityReservation
	ec2:CancelCapacityReservationFleets
	ec2:CancelConversionTask
	ec2:CancelDeclarativePoliciesReport
	ec2:CancelExportTask
	ec2:CancelImageLaunchPermission
	ec2:CancelImportTask
	ec2:CancelReservedInstancesListing
	ec2:CancelSpotFleetRequests
	ec2:CancelSpotInstanceRequests
	ec2:ConfirmProductInstance
	ec2:CopyFpgaImage
	ec2:CopyImage
	ec2:CopySnapshot
	ec2:CopyVolumes
	ec2:CreateCapacityManagerDataExport
	ec2:CreateCapacityReservation

服务前缀	操作
	ec2:CreateCapacityReservationBySplitting
	ec2:CreateCapacityReservationFleet
	ec2:CreateCarrierGateway
	ec2:CreateClientVpnEndpoint
	ec2:CreateClientVpnRoute
	ec2:CreateCoipCidr
	ec2:CreateCoipPool
	ec2:CreateCustomerGateway
	ec2:CreateDefaultSubnet
	ec2:CreateDefaultVpc
	ec2:CreateDelegateMacVolumeOwnershipTask
	ec2:CreateDhcpOptions
	ec2:CreateEgressOnlyInternetGateway
	ec2:CreateFleet
	ec2:CreateFlowLogs
	ec2:CreateFpgaImage
	ec2:CreateImage
	ec2:CreateImageUsageReport
	ec2:CreateInstanceConnectEndpoint
	ec2:CreateInstanceEventWindow
	ec2:CreateInstanceExportTask

服务前缀	操作
	ec2:CreateInternetGateway
	ec2:CreateInterruptibleCapacityReservationAllocation
	ec2:CreateIpam
	ec2:CreateIpamExternalResourceVerificationToken
	ec2:CreateIpamPolicy
	ec2:CreateIpamPool
	ec2:CreateIpamPrefixListResolver
	ec2:CreateIpamPrefixListResolverTarget
	ec2:CreateIpamResourceDiscovery
	ec2:CreateIpamScope
	ec2:CreateKeyPair
	ec2:CreateLaunchTemplateVersion
	ec2:CreateLocalGatewayRoute
	ec2:CreateLocalGatewayRouteTable
	ec2:CreateLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:CreateLocalGatewayRouteTableVpcAssociation
	ec2:CreateLocalGatewayVirtualInterface
	ec2:CreateLocalGatewayVirtualInterfaceGroup
	ec2:CreateMacSystemIntegrityProtectionModificationTask
	ec2:CreateManagedPrefixList

服务前缀	操作
	ec2:CreateNatGateway
	ec2:CreateNetworkAcl
	ec2:CreateNetworkAclEntry
	ec2:CreateNetworkInsightsAccessScope
	ec2:CreateNetworkInsightsPath
	ec2:CreateNetworkInterface
	ec2:CreateNetworkInterfacePermission
	ec2:CreatePlacementGroup
	ec2:CreatePublicIpv4Pool
	ec2:CreateReplaceRootVolumeTask
	ec2:CreateReservedInstancesListing
	ec2:CreateRestoreImageTask
	ec2:CreateRoute
	ec2:CreateRouteServer
	ec2:CreateRouteServerEndpoint
	ec2:CreateRouteServerPeer
	ec2:CreateRouteTable
	ec2:CreateSecurityGroup
	ec2:CreateSnapshots
	ec2:CreateSpotDatafeedSubscription
	ec2:CreateStoreImageTask

服务前缀	操作
	ec2:CreateSubnet
	ec2:CreateSubnetCidrReservation
	ec2:CreateTrafficMirrorFilter
	ec2:CreateTrafficMirrorFilterRule
	ec2:CreateTrafficMirrorSession
	ec2:CreateTrafficMirrorTarget
	ec2:CreateTransitGateway
	ec2:CreateTransitGatewayConnect
	ec2:CreateTransitGatewayConnectPeer
	ec2:CreateTransitGatewayMeteringPolicy
	ec2:CreateTransitGatewayMeteringPolicyEntry
	ec2:CreateTransitGatewayMulticastDomain
	ec2:CreateTransitGatewayPeeringAttachment
	ec2:CreateTransitGatewayPolicyTable
	ec2:CreateTransitGatewayPrefixListReference
	ec2:CreateTransitGatewayRoute
	ec2:CreateTransitGatewayRouteTable
	ec2:CreateTransitGatewayRouteTableAnnouncement
	ec2:CreateTransitGatewayVpcAttachment
	ec2:CreateVerifiedAccessEndpoint
	ec2:CreateVerifiedAccessGroup

服务前缀	操作
	ec2:CreateVerifiedAccessInstance
	ec2:CreateVerifiedAccessTrustProvider
	ec2:CreateVolume
	ec2:CreateVpc
	ec2:CreateVpcBlockPublicAccessExclusion
	ec2:CreateVpcEncryptionControl
	ec2:CreateVpcEndpoint
	ec2:CreateVpcEndpointConnectionNotification
	ec2:CreateVpcEndpointServiceConfiguration
	ec2:CreateVpcPeeringConnection
	ec2:CreateVpnConcentrator
	ec2:CreateVpnConnection
	ec2:CreateVpnConnectionRoute
	ec2:CreateVpnGateway
	ec2:DeleteCapacityManagerDataExport
	ec2:DeleteCarrierGateway
	ec2:DeleteClientVpnEndpoint
	ec2:DeleteClientVpnRoute
	ec2:DeleteCoipCidr
	ec2:DeleteCoipPool
	ec2:DeleteCustomerGateway

服务前缀	操作
	ec2:DeleteDhcpOptions
	ec2:DeleteEgressOnlyInternetGateway
	ec2:DeleteFleets
	ec2:DeleteFlowLogs
	ec2:DeleteFpgaImage
	ec2:DeleteImageUsageReport
	ec2:DeleteInstanceConnectEndpoint
	ec2:DeleteInstanceEventWindow
	ec2:DeleteInternetGateway
	ec2:DeleteIpam
	ec2:DeleteIpamExternalResourceVerificationToken
	ec2:DeleteIpamPolicy
	ec2:DeleteIpamPool
	ec2:DeleteIpamPrefixListResolver
	ec2:DeleteIpamPrefixListResolverTarget
	ec2:DeleteIpamResourceDiscovery
	ec2:DeleteIpamScope
	ec2:DeleteKeyPair
	ec2:DeleteLaunchTemplate
	ec2:DeleteLaunchTemplateVersions
	ec2:DeleteLocalGatewayRoute

服务前缀	操作
	ec2:DeleteLocalGatewayRouteTable
	ec2:DeleteLocalGatewayRouteTableVirtualInterfaceGroupAssociation
	ec2:DeleteLocalGatewayRouteTableVpcAssociation
	ec2:DeleteLocalGatewayVirtualInterface
	ec2:DeleteLocalGatewayVirtualInterfaceGroup
	ec2:DeleteManagedPrefixList
	ec2:DeleteNatGateway
	ec2:DeleteNetworkAcl
	ec2:DeleteNetworkAclEntry
	ec2:DeleteNetworkInsightsAccessScope
	ec2:DeleteNetworkInsightsAccessScopeAnalysis
	ec2:DeleteNetworkInsightsAnalysis
	ec2:DeleteNetworkInsightsPath
	ec2:DeleteNetworkInterface
	ec2:DeleteNetworkInterfacePermission
	ec2:DeletePlacementGroup
	ec2:DeletePublicIpv4Pool
	ec2:DeleteQueuedReservedInstances
	ec2:DeleteRoute
	ec2:DeleteRouteServer

服务前缀	操作
	ec2:DeleteRouteServerEndpoint
	ec2:DeleteRouteServerPeer
	ec2:DeleteRouteTable
	ec2:DeleteSecurityGroup
	ec2:DeleteSpotDatafeedSubscription
	ec2:DeleteSubnet
	ec2:DeleteSubnetCidrReservation
	ec2:DeleteTrafficMirrorFilter
	ec2:DeleteTrafficMirrorFilterRule
	ec2:DeleteTrafficMirrorSession
	ec2:DeleteTrafficMirrorTarget
	ec2:DeleteTransitGateway
	ec2:DeleteTransitGatewayConnect
	ec2:DeleteTransitGatewayConnectPeer
	ec2:DeleteTransitGatewayMeteringPolicy
	ec2:DeleteTransitGatewayMeteringPolicyEntry
	ec2:DeleteTransitGatewayMulticastDomain
	ec2:DeleteTransitGatewayPeeringAttachment
	ec2:DeleteTransitGatewayPolicyTable
	ec2:DeleteTransitGatewayPrefixListReference
	ec2:DeleteTransitGatewayRoute

服务前缀	操作
	ec2:DeleteTransitGatewayRouteTable
	ec2:DeleteTransitGatewayRouteTableAnnouncement
	ec2:DeleteTransitGatewayVpcAttachment
	ec2:DeleteVerifiedAccessEndpoint
	ec2:DeleteVerifiedAccessGroup
	ec2:DeleteVerifiedAccessInstance
	ec2:DeleteVerifiedAccessTrustProvider
	ec2:DeleteVolume
	ec2:DeleteVpc
	ec2:DeleteVpcBlockPublicAccessExclusion
	ec2:DeleteVpcEncryptionControl
	ec2:DeleteVpcEndpointConnectionNotifications
	ec2:DeleteVpcEndpointServiceConfigurations
	ec2:DeleteVpcEndpoints
	ec2:DeleteVpcPeeringConnection
	ec2:DeleteVpnConcentrator
	ec2:DeleteVpnConnection
	ec2:DeleteVpnConnectionRoute
	ec2:DeleteVpnGateway
	ec2:DeprovisionByoipCidr
	ec2:DeprovisionIpamByoasn

服务前缀	操作
	ec2:DeprovisionIpamPoolCidr
	ec2:DeprovisionPublicIpv4PoolCidr
	ec2:DeregisterImage
	ec2:DeregisterInstanceEventNotificationAttributes
	ec2:DeregisterTransitGatewayMulticastGroupMembers
	ec2:DeregisterTransitGatewayMulticastGroupSources
	ec2:DescribeAccountAttributes
	ec2:DescribeAddressTransfers
	ec2:DescribeAddresses
	ec2:DescribeAddressesAttribute
	ec2:DescribeAggregateIdFormat
	ec2:DescribeAvailabilityZones
	ec2:DescribeAwsNetworkPerformanceMetricSubscriptions
	ec2:DescribeBundleTasks
	ec2:DescribeByoipCidrs
	ec2:DescribeCapacityBlockExtensionHistory
	ec2:DescribeCapacityBlockExtensionOfferings
	ec2:DescribeCapacityBlockStatus
	ec2:DescribeCapacityBlocks
	ec2:DescribeCapacityManagerDataExports
	ec2:DescribeCapacityReservationBillingRequests

服务前缀	操作
	ec2:DescribeCapacityReservationFleets
	ec2:DescribeCapacityReservationTopology
	ec2:DescribeCapacityReservations
	ec2:DescribeCarrierGateways
	ec2:DescribeClassicLinkInstances
	ec2:DescribeClientVpnAuthorizationRules
	ec2:DescribeClientVpnConnections
	ec2:DescribeClientVpnEndpoints
	ec2:DescribeClientVpnRoutes
	ec2:DescribeClientVpnTargetNetworks
	ec2:DescribeCoipPools
	ec2:DescribeConversionTasks
	ec2:DescribeCustomerGateways
	ec2:DescribeDeclarativePoliciesReports
	ec2:DescribeDhcpOptions
	ec2:DescribeEgressOnlyInternetGateways
	ec2:DescribeElasticGpus
	ec2:DescribeExportImageTasks
	ec2:DescribeExportTasks
	ec2:DescribeFastLaunchImages
	ec2:DescribeFastSnapshotRestores

服务前缀	操作
	ec2:DescribeFleetHistory
	ec2:DescribeFleetInstances
	ec2:DescribeFleets
	ec2:DescribeFlowLogs
	ec2:DescribeFpgaImageAttribute
	ec2:DescribeFpgaImages
	ec2:DescribeHostReservationOfferings
	ec2:DescribeHostReservations
	ec2:DescribeHosts
	ec2:DescribeIamInstanceProfileAssociations
	ec2:DescribeIdFormat
	ec2:DescribeIdentityIdFormat
	ec2:DescribeImageAttribute
	ec2:DescribeImageReferences
	ec2:DescribeImageUsageReportEntries
	ec2:DescribeImageUsageReports
	ec2:DescribeImportImageTasks
	ec2:DescribeImportSnapshotTasks
	ec2:DescribeInstanceConnectEndpoints
	ec2:DescribeInstanceCreditSpecifications
	ec2:DescribeInstanceEventNotificationAttributes

服务前缀	操作
	ec2:DescribeInstanceEventWindows
	ec2:DescribeInstanceImageMetadata
	ec2:DescribeInstanceSqlHaHistoryStates
	ec2:DescribeInstanceSqlHaStates
	ec2:DescribeInstanceTopology
	ec2:DescribeInstanceTypes
	ec2:DescribeInternetGateways
	ec2:DescribeIpamByoasn
	ec2:DescribeIpamExternalResourceVerificationTokens
	ec2:DescribeIpamPolicies
	ec2:DescribeIpamPools
	ec2:DescribeIpamPrefixListResolverTargets
	ec2:DescribeIpamPrefixListResolvers
	ec2:DescribeIpamResourceDiscoveries
	ec2:DescribeIpamResourceDiscoveryAssociations
	ec2:DescribeIpamScopes
	ec2:DescribeIpams
	ec2:DescribeIpv6Pools
	ec2:DescribeKeyPairs
	ec2:DescribeLocalGatewayRouteTableVirtualInterfaceGroupAssociations

服务前缀	操作
	ec2:DescribeLocalGatewayRouteTableVpcAssociations
	ec2:DescribeLocalGatewayRouteTables
	ec2:DescribeLocalGatewayVirtualInterfaceGroups
	ec2:DescribeLocalGatewayVirtualInterfaces
	ec2:DescribeLocalGateways
	ec2:DescribeLockedSnapshots
	ec2:DescribeMacHosts
	ec2:DescribeMacModificationTasks
	ec2:DescribeManagedPrefixLists
	ec2:DescribeMovingAddresses
	ec2:DescribeNatGateways
	ec2:DescribeNetworkAcls
	ec2:DescribeNetworkInsightsAccessScopeAnalyses
	ec2:DescribeNetworkInsightsAccessScopes
	ec2:DescribeNetworkInsightsAnalyses
	ec2:DescribeNetworkInsightsPaths
	ec2:DescribeNetworkInterfaceAttribute
	ec2:DescribeNetworkInterfacePermissions
	ec2:DescribeNetworkInterfaces
	ec2:DescribeOutpostLags
	ec2:DescribePlacementGroups

服务前缀	操作
	ec2:DescribePrefixLists
	ec2:DescribePrincipalIdFormat
	ec2:DescribePublicIpv4Pools
	ec2:DescribeRegions
	ec2:DescribeReplaceRootVolumeTasks
	ec2:DescribeReservedInstances
	ec2:DescribeReservedInstancesListings
	ec2:DescribeReservedInstancesModifications
	ec2:DescribeReservedInstancesOfferings
	ec2:DescribeRouteServerEndpoints
	ec2:DescribeRouteServerPeers
	ec2:DescribeRouteServers
	ec2:DescribeRouteTables
	ec2:DescribeScheduledInstanceAvailability
	ec2:DescribeScheduledInstances
	ec2:DescribeSecurityGroupReferences
	ec2:DescribeSecurityGroupRules
	ec2:DescribeSecurityGroupVpcAssociations
	ec2:DescribeSecurityGroups
	ec2:DescribeServiceLinkVirtualInterfaces
	ec2:DescribeSnapshotAttribute

服务前缀	操作
	ec2:DescribeSnapshotTierStatus
	ec2:DescribeSpotDatafeedSubscription
	ec2:DescribeSpotFleetInstances
	ec2:DescribeSpotFleetRequestHistory
	ec2:DescribeSpotFleetRequests
	ec2:DescribeSpotInstanceRequests
	ec2:DescribeSpotPriceHistory
	ec2:DescribeStaleSecurityGroups
	ec2:DescribeStoreImageTasks
	ec2:DescribeTrafficMirrorFilterRules
	ec2:DescribeTrafficMirrorFilters
	ec2:DescribeTrafficMirrorSessions
	ec2:DescribeTrafficMirrorTargets
	ec2:DescribeTransitGatewayAttachments
	ec2:DescribeTransitGatewayConnectPeers
	ec2:DescribeTransitGatewayConnects
	ec2:DescribeTransitGatewayMeteringPolicies
	ec2:DescribeTransitGatewayMulticastDomains
	ec2:DescribeTransitGatewayPeeringAttachments
	ec2:DescribeTransitGatewayPolicyTables
	ec2:DescribeTransitGatewayRouteTableAnnouncements

服务前缀	操作
	ec2:DescribeTransitGatewayRouteTables
	ec2:DescribeTransitGatewayVpcAttachments
	ec2:DescribeTransitGateways
	ec2:DescribeTrunkInterfaceAssociations
	ec2:DescribeVerifiedAccessEndpoints
	ec2:DescribeVerifiedAccessGroups
	ec2:DescribeVerifiedAccessInstanceLoggingConfigurations
	ec2:DescribeVerifiedAccessInstances
	ec2:DescribeVerifiedAccessTrustProviders
	ec2:DescribeVolumeAttribute
	ec2:DescribeVolumeStatus
	ec2:DescribeVolumes
	ec2:DescribeVolumesModifications
	ec2:DescribeVpcAttribute
	ec2:DescribeVpcBlockPublicAccessExclusions
	ec2:DescribeVpcBlockPublicAccessOptions
	ec2:DescribeVpcClassicLink
	ec2:DescribeVpcClassicLinkDnsSupport
	ec2:DescribeVpcEncryptionControls
	ec2:DescribeVpcEndpointAssociations
	ec2:DescribeVpcEndpointConnectionNotifications

服务前缀	操作
	ec2:DescribeVpcEndpointConnections
	ec2:DescribeVpcEndpointServiceConfigurations
	ec2:DescribeVpcEndpointServicePermissions
	ec2:DescribeVpcEndpointServices
	ec2:DescribeVpcEndpoints
	ec2:DescribeVpcPeeringConnections
	ec2:DescribeVpcs
	ec2:DescribeVpnConcentrators
	ec2:DescribeVpnConnections
	ec2:DescribeVpnGateways
	ec2:DetachClassicLinkVpc
	ec2:DetachInternetGateway
	ec2:DetachNetworkInterface
	ec2:DetachVerifiedAccessTrustProvider
	ec2:DetachVolume
	ec2:DetachVpnGateway
	ec2:DisableAddressTransfer
	ec2:DisableAllowedImagesSettings
	ec2:DisableAwsNetworkPerformanceMetricSubscription
	ec2:DisableCapacityManager
	ec2:DisableEbsEncryptionByDefault

服务前缀	操作
	ec2:DisableFastLaunch
	ec2:DisableFastSnapshotRestores
	ec2:DisableImage
	ec2:DisableImageBlockPublicAccess
	ec2:DisableImageDeprecation
	ec2:DisableImageDeregistrationProtection
	ec2:DisableInstanceSqlHaStandbyDetections
	ec2:DisableIpamOrganizationAdminAccount
	ec2:DisableIpamPolicy
	ec2:DisableRouteServerPropagation
	ec2:DisableSerialConsoleAccess
	ec2:DisableSnapshotBlockPublicAccess
	ec2:DisableTransitGatewayRouteTablePropagation
	ec2:DisableVgwRoutePropagation
	ec2:DisableVpcClassicLink
	ec2:DisableVpcClassicLinkDnsSupport
	ec2:DisassociateAddress
	ec2:DisassociateCapacityReservationBillingOwner
	ec2:DisassociateClientVpnTargetNetwork
	ec2:DisassociateEnclaveCertificateIamRole
	ec2:DisassociateIamInstanceProfile

服务前缀	操作
	ec2:DisassociateInstanceEventWindow
	ec2:DisassociateIpamByoasn
	ec2:DisassociateIpamResourceDiscovery
	ec2:DisassociateNatGatewayAddress
	ec2:DisassociateRouteServer
	ec2:DisassociateRouteTable
	ec2:DisassociateSecurityGroupVpc
	ec2:DisassociateSubnetCidrBlock
	ec2:DisassociateTransitGatewayMulticastDomain
	ec2:DisassociateTransitGatewayPolicyTable
	ec2:DisassociateTransitGatewayRouteTable
	ec2:DisassociateTrunkInterface
	ec2:DisassociateVpcCidrBlock
	ec2:EnableAddressTransfer
	ec2:EnableAllowedImagesSettings
	ec2:EnableAwsNetworkPerformanceMetricSubscription
	ec2:EnableCapacityManager
	ec2:EnableEbsEncryptionByDefault
	ec2:EnableFastLaunch
	ec2:EnableFastSnapshotRestores
	ec2:EnableImage

服务前缀	操作
	ec2:EnableImageBlockPublicAccess
	ec2:EnableImageDeprecation
	ec2:EnableImageDeregistrationProtection
	ec2:EnableInstanceSqlHaStandbyDetections
	ec2:EnableIpamOrganizationAdminAccount
	ec2:EnableIpamPolicy
	ec2:EnableReachabilityAnalyzerOrganizationSharing
	ec2:EnableRouteServerPropagation
	ec2:EnableSerialConsoleAccess
	ec2:EnableSnapshotBlockPublicAccess
	ec2:EnableTransitGatewayRouteTablePropagation
	ec2:EnableVgwRoutePropagation
	ec2:EnableVolumeIO
	ec2:EnableVpcClassicLink
	ec2:EnableVpcClassicLinkDnsSupport
	ec2:ExportClientVpnClientCertificateRevocationList
	ec2:ExportClientVpnClientConfiguration
	ec2:ExportImage
	ec2:ExportTransitGatewayRoutes
	ec2:ExportVerifiedAccessInstanceClientConfiguration
	ec2:GetActiveVpnTunnelStatus

服务前缀	操作
	ec2:GetAllowedImagesSettings
	ec2:GetAssociatedEnclaveCertificateIamRoles
	ec2:GetAssociatedIpv6PoolCidrs
	ec2:GetAwsNetworkPerformanceData
	ec2:GetCapacityManagerAttributes
	ec2:GetCapacityManagerMetricData
	ec2:GetCapacityManagerMetricDimensions
	ec2:GetCapacityReservationUsage
	ec2:GetCoipPoolUsage
	ec2:GetConsoleOutput
	ec2:GetConsoleScreenshot
	ec2:GetDeclarativePoliciesReportSummary
	ec2:GetDefaultCreditSpecification
	ec2:GetEbsDefaultKmsKeyId
	ec2:GetEbsEncryptionByDefault
	ec2:GetEnabledIpamPolicy
	ec2:GetFlowLogsIntegrationTemplate
	ec2:GetGroupsForCapacityReservation
	ec2:GetHostReservationPurchasePreview
	ec2:GetImageAncestry
	ec2:GetImageBlockPublicAccessState

服务前缀	操作
	ec2:GetInstanceMetadataDefaults
	ec2:GetInstanceTpmEkPub
	ec2:GetInstanceTypesFromInstanceRequirements
	ec2:GetInstanceUefiData
	ec2:GetIpamAddressHistory
	ec2:GetIpamDiscoveredAccounts
	ec2:GetIpamDiscoveredPublicAddresses
	ec2:GetIpamDiscoveredResourceCidrs
	ec2:GetIpamPolicyAllocationRules
	ec2:GetIpamPolicyOrganizationTargets
	ec2:GetIpamPoolAllocations
	ec2:GetIpamPoolCidrs
	ec2:GetIpamPrefixListResolverRules
	ec2:GetIpamPrefixListResolverVersionEntries
	ec2:GetIpamPrefixListResolverVersions
	ec2:GetIpamResourceCidrs
	ec2:GetLaunchTemplateData
	ec2:GetManagedPrefixListAssociations
	ec2:GetManagedPrefixListEntries
	ec2:GetNetworkInsightsAccessScopeAnalysisFindings
	ec2:GetNetworkInsightsAccessScopeContent

服务前缀	操作
	ec2:GetPasswordData
	ec2:GetReservedInstancesExchangeQuote
	ec2:GetRouteServerAssociations
	ec2:GetRouteServerPropagations
	ec2:GetRouteServerRoutingDatabase
	ec2:GetSecurityGroupsForVpc
	ec2:GetSerialConsoleAccessStatus
	ec2:GetSnapshotBlockPublicAccessState
	ec2:GetSpotPlacementScores
	ec2:GetSubnetCidrReservations
	ec2:GetTransitGatewayAttachmentPropagations
	ec2:GetTransitGatewayMeteringPolicyEntries
	ec2:GetTransitGatewayMulticastDomainAssociations
	ec2:GetTransitGatewayPolicyTableAssociations
	ec2:GetTransitGatewayPolicyTableEntries
	ec2:GetTransitGatewayPrefixListReferences
	ec2:GetTransitGatewayRouteTableAssociations
	ec2:GetTransitGatewayRouteTablePropagations
	ec2:GetVerifiedAccessEndpointPolicy
	ec2:GetVerifiedAccessEndpointTargets
	ec2:GetVerifiedAccessGroupPolicy

服务前缀	操作
	ec2:GetVpcResourcesBlockingEncryptionEnforcement
	ec2:GetVpnConnectionDeviceSampleConfiguration
	ec2:GetVpnConnectionDeviceTypes
	ec2:GetVpnTunnelReplacementStatus
	ec2:ImportClientVpnClientCertificateRevocationList
	ec2:ImportImage
	ec2:ImportInstance
	ec2:ImportKeyPair
	ec2:ImportSnapshot
	ec2:ImportVolume
	ec2:ListImagesInRecycleBin
	ec2:ListSnapshotsInRecycleBin
	ec2:ListVolumesInRecycleBin
	ec2:LockSnapshot
	ec2:ModifyAddressAttribute
	ec2:ModifyAvailabilityZoneGroup
	ec2:ModifyCapacityReservation
	ec2:ModifyCapacityReservationFleet
	ec2:ModifyClientVpnEndpoint
	ec2:ModifyDefaultCreditSpecification
	ec2:ModifyEbsDefaultKmsKeyId

服务前缀	操作
	ec2:ModifyFleet
	ec2:ModifyFpgaImageAttribute
	ec2:ModifyHosts
	ec2:ModifyIdFormat
	ec2:ModifyIdentityIdFormat
	ec2:ModifyImageAttribute
	ec2:ModifyInstanceAttribute
	ec2:ModifyInstanceCapacityReservationAttributes
	ec2:ModifyInstanceConnectEndpoint
	ec2:ModifyInstanceCpuOptions
	ec2:ModifyInstanceCreditSpecification
	ec2:ModifyInstanceEventStartTime
	ec2:ModifyInstanceEventWindow
	ec2:ModifyInstanceMaintenanceOptions
	ec2:ModifyInstanceMetadataDefaults
	ec2:ModifyInstanceMetadataOptions
	ec2:ModifyInstanceNetworkPerformanceOptions
	ec2:ModifyInstancePlacement
	ec2:ModifyIpam
	ec2:ModifyIpamPolicyAllocationRules
	ec2:ModifyIpamPool

服务前缀	操作
	ec2:ModifyIpamPrefixListResolver
	ec2:ModifyIpamPrefixListResolverTarget
	ec2:ModifyIpamResourceCidr
	ec2:ModifyIpamResourceDiscovery
	ec2:ModifyIpamScope
	ec2:ModifyLaunchTemplate
	ec2:ModifyLocalGatewayRoute
	ec2:ModifyManagedPrefixList
	ec2:ModifyNetworkInterfaceAttribute
	ec2:ModifyPrivateDnsNameOptions
	ec2:ModifyPublicIpDnsNameOptions
	ec2:ModifyReservedInstances
	ec2:ModifyRouteServer
	ec2:ModifySecurityGroupRules
	ec2:ModifySnapshotAttribute
	ec2:ModifySnapshotTier
	ec2:ModifySpotFleetRequest
	ec2:ModifySubnetAttribute
	ec2:ModifyTrafficMirrorFilterNetworkServices
	ec2:ModifyTrafficMirrorFilterRule
	ec2:ModifyTrafficMirrorSession

服务前缀	操作
	ec2:ModifyTransitGateway
	ec2:ModifyTransitGatewayMeteringPolicy
	ec2:ModifyTransitGatewayPrefixListReference
	ec2:ModifyTransitGatewayVpcAttachment
	ec2:ModifyVerifiedAccessEndpoint
	ec2:ModifyVerifiedAccessEndpointPolicy
	ec2:ModifyVerifiedAccessGroup
	ec2:ModifyVerifiedAccessGroupPolicy
	ec2:ModifyVerifiedAccessInstance
	ec2:ModifyVerifiedAccessInstanceLoggingConfiguration
	ec2:ModifyVerifiedAccessTrustProvider
	ec2:ModifyVolume
	ec2:ModifyVolumeAttribute
	ec2:ModifyVpcAttribute
	ec2:ModifyVpcBlockPublicAccessExclusion
	ec2:ModifyVpcBlockPublicAccessOptions
	ec2:ModifyVpcEncryptionControl
	ec2:ModifyVpcEndpoint
	ec2:ModifyVpcEndpointConnectionNotification
	ec2:ModifyVpcEndpointServiceConfiguration
	ec2:ModifyVpcEndpointServicePayerResponsibility

服务前缀	操作
	ec2:ModifyVpcEndpointServicePermissions
	ec2:ModifyVpcPeeringConnectionOptions
	ec2:ModifyVpcTenancy
	ec2:ModifyVpnConnection
	ec2:ModifyVpnConnectionOptions
	ec2:ModifyVpnTunnelCertificate
	ec2:ModifyVpnTunnelOptions
	ec2:MonitorInstances
	ec2:MoveAddressToVpc
	ec2:MoveByoipCidrToIpam
	ec2:MoveCapacityReservationInstances
	ec2:ProvisionByoipCidr
	ec2:ProvisionIpamByoasn
	ec2:ProvisionIpamPoolCidr
	ec2:ProvisionPublicIpv4PoolCidr
	ec2:PurchaseCapacityBlockExtension
	ec2:PurchaseHostReservation
	ec2:PurchaseReservedInstancesOffering
	ec2:PurchaseScheduledInstances
	ec2:RebootInstances
	ec2:RegisterImage

服务前缀	操作
	ec2:RegisterInstanceEventNotificationAttributes
	ec2:RegisterTransitGatewayMulticastGroupMembers
	ec2:RegisterTransitGatewayMulticastGroupSources
	ec2:RejectCapacityReservationBillingOwnership
	ec2:RejectTransitGatewayMulticastDomainAssociations
	ec2:RejectTransitGatewayPeeringAttachment
	ec2:RejectTransitGatewayVpcAttachment
	ec2:RejectVpcEndpointConnections
	ec2:RejectVpcPeeringConnection
	ec2:ReleaseAddress
	ec2:ReleaseHosts
	ec2:ReleaseIpamPoolAllocation
	ec2:ReplaceIamInstanceProfileAssociation
	ec2:ReplaceImageCriteriaInAllowedImagesSettings
	ec2:ReplaceNetworkAclAssociation
	ec2:ReplaceNetworkAclEntry
	ec2:ReplaceRoute
	ec2:ReplaceRouteTableAssociation
	ec2:ReplaceTransitGatewayRoute
	ec2:ReplaceVpnTunnel
	ec2:ReportInstanceStatus

服务前缀	操作
	ec2:RequestSpotFleet
	ec2:RequestSpotInstances
	ec2:ResetAddressAttribute
	ec2:ResetEbsDefaultKmsKeyId
	ec2:ResetFpgaImageAttribute
	ec2:ResetImageAttribute
	ec2:ResetInstanceAttribute
	ec2:ResetNetworkInterfaceAttribute
	ec2:ResetSnapshotAttribute
	ec2:RestoreAddressToClassic
	ec2:RestoreImageFromRecycleBin
	ec2:RestoreManagedPrefixListVersion
	ec2:RestoreSnapshotFromRecycleBin
	ec2:RestoreSnapshotTier
	ec2:RestoreVolumeFromRecycleBin
	ec2:RevokeClientVpnIngress
	ec2:RevokeSecurityGroupEgress
	ec2:RevokeSecurityGroupIngress
	ec2:RunInstances
	ec2:RunScheduledInstances
	ec2:SearchLocalGatewayRoutes

服务前缀	操作
	ec2:SearchTransitGatewayMulticastGroups
	ec2:SearchTransitGatewayRoutes
	ec2:SendDiagnosticInterrupt
	ec2:StartDeclarativePoliciesReport
	ec2:StartInstances
	ec2:StartNetworkInsightsAccessScopeAnalysis
	ec2:StartNetworkInsightsAnalysis
	ec2:StartVpcEndpointServicePrivateDnsVerification
	ec2:TerminateClientVpnConnections
	ec2:UnassignIpv6Addresses
	ec2:UnassignPrivateIpAddresses
	ec2:UnassignPrivateNatGatewayAddress
	ec2:UnlockSnapshot
	ec2:UnmonitorInstances
	ec2:UpdateCapacityManagerOrganizationsAccess
	ec2:UpdateInterruptibleCapacityReservationAllocation
	ec2:UpdateSecurityGroupRuleDescriptionsEgress
	ec2:UpdateSecurityGroupRuleDescriptionsIngress
	ec2:WithdrawByoipCidr

服务前缀	操作
ecr	ecr:BatchCheckLayerAvailability ecr:BatchDeleteImage ecr:BatchGetImage ecr:BatchGetRepositoryScanningConfiguration ecr:CompleteLayerUpload ecr:CreatePullThroughCacheRule ecr:CreateRepositoryCreationTemplate ecr:DeleteLifecyclePolicy ecr:DeletePullThroughCacheRule ecr:DeleteRegistryPolicy ecr:DeleteRepository ecr:DeleteRepositoryCreationTemplate ecr:DeleteRepositoryPolicy ecr:DeleteSigningConfiguration ecr:DescribeImageReplicationStatus ecr:DescribeImageScanFindings ecr:DescribeImages ecr:DescribePullThroughCacheRules ecr:DescribeRegistry ecr:DescribeRepositories ecr:DescribeRepositoryCreationTemplates

服务前缀	操作
	ecr:GetAccountSetting ecr:GetAuthorizationToken ecr:GetDownloadUrlForLayer ecr:GetLifecyclePolicy ecr:GetLifecyclePolicyPreview ecr:GetRegistryPolicy ecr:GetRegistryScanningConfiguration ecr:GetRepositoryPolicy ecr:GetSigningConfiguration ecr:InitiateLayerUpload ecr:ListImages ecr:ListPullTimeUpdateExclusions ecr:PutAccountSetting ecr:PutImage ecr:PutImageScanningConfiguration ecr:PutRegistryPolicy ecr:PutRegistryScanningConfiguration ecr:PutReplicationConfiguration ecr:StartImageScan ecr:StartLifecyclePolicyPreview ecr:UpdatePullThroughCacheRule

服务前缀	操作
	ecr:UpdateRepositoryCreationTemplate ecr:UploadLayerPart ecr:ValidatePullThroughCacheRule

服务前缀	操作
ecr-public	ecr-public:BatchCheckLayerAvailability
	ecr-public:BatchDeleteImage
	ecr-public:CompleteLayerUpload
	ecr-public:CreateRepository
	ecr-public:DeleteRepository
	ecr-public:DeleteRepositoryPolicy
	ecr-public:DescribeImages
	ecr-public:DescribeRegistries
	ecr-public:DescribeRepositories
	ecr-public:GetAuthorizationToken
	ecr-public:GetRegistryCatalogData
	ecr-public:GetRepositoryCatalogData
	ecr-public:GetRepositoryPolicy
	ecr-public:InitiateLayerUpload
	ecr-public:PutImage
	ecr-public:PutRegistryCatalogData
	ecr-public:PutRepositoryCatalogData
	ecr-public:SetRepositoryPolicy
	ecr-public:UploadLayerPart

服务前缀	操作
ecs	ecs:CreateCapacityProvider ecs:CreateCluster ecs:CreateService ecs:CreateTaskSet ecs:DeleteAccountSetting ecs:DeleteAttributes ecs:DeleteCapacityProvider ecs:DeleteCluster ecs:DeleteExpressGatewayService ecs:DeleteService ecs:DeleteTaskDefinitions ecs:DeleteTaskSet ecs:DeregisterContainerInstance ecs:DeregisterTaskDefinition ecs:DescribeCapacityProviders ecs:DescribeClusters ecs:DescribeContainerInstances ecs:DescribeExpressGatewayService ecs:DescribeServiceDeployments ecs:DescribeServiceRevisions ecs:DescribeServices

服务前缀	操作
	ecs:DescribeTaskDefinition ecs:DescribeTaskSets ecs:DescribeTasks ecs:DiscoverPollEndpoint ecs:ExecuteCommand ecs:GetTaskProtection ecs:ListAccountSettings ecs:ListAttributes ecs:ListClusters ecs:ListContainerInstances ecs:ListServiceDeployments ecs:ListServices ecs:ListServicesByNamespace ecs:ListTaskDefinitionFamilies ecs:ListTaskDefinitions ecs:ListTasks ecs:PutAccountSetting ecs:PutAccountSettingDefault ecs:PutAttributes ecs:PutClusterCapacityProviders ecs:RegisterContainerInstance

服务前缀	操作
	ecs:RunTask ecs:StartTask ecs:StopServiceDeployment ecs:StopTask ecs:SubmitAttachmentStateChanges ecs:SubmitContainerStateChange ecs:SubmitTaskStateChange ecs:UpdateCapacityProvider ecs:UpdateCluster ecs:UpdateClusterSettings ecs:UpdateContainerAgent ecs:UpdateContainerInstancesState ecs:UpdateExpressGatewayService ecs:UpdateService ecs:UpdateServicePrimaryTaskSet ecs:UpdateTaskProtection ecs:UpdateTaskSet

服务前缀	操作
eks	eks:AssociateAccessPolicy
	eks:AssociateEncryptionConfig
	eks:AssociateIdentityProviderConfig
	eks:CreateAccessEntry
	eks:CreateAddon
	eks:CreateCluster
	eks:CreateEksAnywhereSubscription
	eks:CreateFargateProfile
	eks:CreateNodegroup
	eks:DeleteAccessEntry
	eks:DeleteAddon
	eks:DeleteCapability
	eks:DeleteCluster
	eks:DeleteEksAnywhereSubscription
	eks:DeleteFargateProfile
	eks:DeleteNodegroup
	eks:DeletePodIdentityAssociation
	eks:DeregisterCluster
	eks:DescribeAccessEntry
	eks:DescribeAddon
	eks:DescribeAddonConfiguration

服务前缀	操作
	eks:DescribeAddonVersions
	eks:DescribeCapability
	eks:DescribeCluster
	eks:DescribeClusterVersions
	eks:DescribeEksAnywhereSubscription
	eks:DescribeFargateProfile
	eks:DescribeIdentityProviderConfig
	eks:DescribeInsight
	eks:DescribeInsightsRefresh
	eks:DescribeNodegroup
	eks:DescribePodIdentityAssociation
	eks:DescribeUpdate
	eks:DisassociateAccessPolicy
	eks:DisassociateIdentityProviderConfig
	eks:ListAccessEntries
	eks:ListAccessPolicies
	eks:ListAddons
	eks:ListAssociatedAccessPolicies
	eks:ListCapabilities
	eks:ListClusters
	eks:ListEksAnywhereSubscriptions

服务前缀	操作
	eks:ListFargateProfiles
	eks:ListIdentityProviderConfigs
	eks:ListInsights
	eks:ListNodegroups
	eks:ListPodIdentityAssociations
	eks:ListUpdates
	eks:RegisterCluster
	eks:StartInsightsRefresh
	eks:UpdateAccessEntry
	eks:UpdateAddon
	eks:UpdateCapability
	eks:UpdateClusterConfig
	eks:UpdateClusterVersion
	eks:UpdateEksAnywhereSubscription
	eks:UpdateNodegroupConfig
	eks:UpdateNodegroupVersion
	eks:UpdatePodIdentityAssociation

服务前缀	操作
elasticache	elasticache:AuthorizeCacheSecurityGroupIngress elasticache:BatchApplyUpdateAction elasticache:BatchStopUpdateAction elasticache:CompleteMigration elasticache:CopyServerlessCacheSnapshot elasticache:CopySnapshot elasticache:CreateCacheCluster elasticache:CreateCacheParameterGroup elasticache:CreateCacheSecurityGroup elasticache:CreateCacheSubnetGroup elasticache:CreateGlobalReplicationGroup elasticache:CreateReplicationGroup elasticache:CreateServerlessCache elasticache:CreateServerlessCacheSnapshot elasticache:CreateSnapshot elasticache:CreateUser elasticache:CreateUserGroup elasticache:DecreaseNodeGroupsInGlobalReplicationGroup elasticache:DecreaseReplicaCount elasticache:DeleteCacheCluster elasticache:DeleteCacheParameterGroup

服务前缀	操作
	elasticache:DeleteCacheSecurityGroup elasticache:DeleteCacheSubnetGroup elasticache:DeleteGlobalReplicationGroup elasticache:DeleteReplicationGroup elasticache:DeleteServerlessCache elasticache:DeleteServerlessCacheSnapshot elasticache:DeleteSnapshot elasticache:DeleteUser elasticache:DeleteUserGroup elasticache:DescribeCacheClusters elasticache:DescribeCacheEngineVersions elasticache:DescribeCacheParameterGroups elasticache:DescribeCacheParameters elasticache:DescribeCacheSecurityGroups elasticache:DescribeCacheSubnetGroups elasticache:DescribeEngineDefaultParameters elasticache:DescribeEvents elasticache:DescribeGlobalReplicationGroups elasticache:DescribeReplicationGroups elasticache:DescribeReservedCacheNodes elasticache:DescribeReservedCacheNodesOfferings

服务前缀	操作
	elasticache:DescribeServerlessCacheSnapshots elasticache:DescribeServerlessCaches elasticache:DescribeServiceUpdates elasticache:DescribeSnapshots elasticache:DescribeUpdateActions elasticache:DescribeUserGroups elasticache:DescribeUsers elasticache:DisassociateGlobalReplicationGroup elasticache:ExportServerlessCacheSnapshot elasticache:FailoverGlobalReplicationGroup elasticache:IncreaseNodeGroupsInGlobalReplicationGroup elasticache:IncreaseReplicaCount elasticache:ListAllowedNodeTypeModifications elasticache:ModifyCacheCluster elasticache:ModifyCacheParameterGroup elasticache:ModifyCacheSubnetGroup elasticache:ModifyGlobalReplicationGroup elasticache:ModifyReplicationGroup elasticache:ModifyReplicationGroupShardConfiguration elasticache:ModifyServerlessCache elasticache:ModifyUser

服务前缀	操作
	elasticache:ModifyUserGroup elasticache:PurchaseReservedCacheNodesOffering elasticache:RebalanceSlotsInGlobalReplicationGroup elasticache:RebootCacheCluster elasticache:ResetCacheParameterGroup elasticache:RevokeCacheSecurityGroupIngress elasticache:StartMigration elasticache:TestFailover elasticache:TestMigration

服务前缀	操作
elasticbeanstalk	elasticbeanstalk:AbortEnvironmentUpdate
	elasticbeanstalk:ApplyEnvironmentManagedAction
	elasticbeanstalk:AssociateEnvironmentOperationsRole
	elasticbeanstalk:CheckDNSAvailability
	elasticbeanstalk:ComposeEnvironments
	elasticbeanstalk:CreateApplication
	elasticbeanstalk:CreateApplicationVersion
	elasticbeanstalk:CreateConfigurationTemplate
	elasticbeanstalk:CreateEnvironment
	elasticbeanstalk:CreatePlatformVersion
	elasticbeanstalk:CreateStorageLocation
	elasticbeanstalk:DeleteApplication
	elasticbeanstalk:DeleteApplicationVersion
	elasticbeanstalk:DeleteConfigurationTemplate
	elasticbeanstalk:DeleteEnvironmentConfiguration
	elasticbeanstalk:DeletePlatformVersion
	elasticbeanstalk:DescribeAccountAttributes
	elasticbeanstalk:DescribeApplicationVersions
	elasticbeanstalk:DescribeApplications
	elasticbeanstalk:DescribeConfigurationOptions
	elasticbeanstalk:DescribeConfigurationSettings

服务前缀	操作
	elasticbeanstalk:DescribeEnvironmentHealth
	elasticbeanstalk:DescribeEnvironmentManagedActionHistory
	elasticbeanstalk:DescribeEnvironmentManagedActions
	elasticbeanstalk:DescribeEnvironmentResources
	elasticbeanstalk:DescribeEnvironments
	elasticbeanstalk:DescribeEvents
	elasticbeanstalk:DescribeInstancesHealth
	elasticbeanstalk:DescribePlatformVersion
	elasticbeanstalk:DisassociateEnvironmentOperationsRole
	elasticbeanstalk:ListAvailableSolutionStacks
	elasticbeanstalk:ListPlatformBranches
	elasticbeanstalk:ListPlatformVersions
	elasticbeanstalk:RebuildEnvironment
	elasticbeanstalk:RequestEnvironmentInfo
	elasticbeanstalk:RestartAppServer
	elasticbeanstalk:RetrieveEnvironmentInfo
	elasticbeanstalk:SwapEnvironmentCNAMEs
	elasticbeanstalk:TerminateEnvironment
	elasticbeanstalk:UpdateApplication
	elasticbeanstalk:UpdateApplicationResourceLifecycle
	elasticbeanstalk:UpdateApplicationVersion

服务前缀	操作
	elasticbeanstalk:UpdateConfigurationTemplate elasticbeanstalk:UpdateEnvironment elasticbeanstalk:ValidateConfigurationSettings

服务前缀	操作
elasticfilesystem	elasticfilesystem:CreateAccessPoint elasticfilesystem:CreateFileSystem elasticfilesystem:CreateMountTarget elasticfilesystem:CreateReplicationConfiguration elasticfilesystem:DeleteAccessPoint elasticfilesystem:DeleteFileSystem elasticfilesystem:DeleteFileSystemPolicy elasticfilesystem:DeleteMountTarget elasticfilesystem:DeleteReplicationConfiguration elasticfilesystem:DescribeAccessPoints elasticfilesystem:DescribeAccountPreferences elasticfilesystem:DescribeBackupPolicy elasticfilesystem:DescribeFileSystemPolicy elasticfilesystem:DescribeFileSystems elasticfilesystem:DescribeLifecycleConfiguration elasticfilesystem:DescribeMountTargetSecurityGroups elasticfilesystem:DescribeMountTargets elasticfilesystem:DescribeReplicationConfigurations elasticfilesystem:ModifyMountTargetSecurityGroups elasticfilesystem:PutAccountPreferences elasticfilesystem:PutBackupPolicy

服务前缀	操作
	elasticfilesystem:PutFileSystemPolicy elasticfilesystem:PutLifecycleConfiguration elasticfilesystem:UpdateFileSystem elasticfilesystem:UpdateFileSystemProtection

服务前缀	操作
elasticloadbalancing	elasticloadbalancing:AddListenerCertificates
	elasticloadbalancing:AddTrustStoreRevocations
	elasticloadbalancing:ApplySecurityGroupsToLoadBalancer
	elasticloadbalancing:AttachLoadBalancerToSubnets
	elasticloadbalancing:ConfigureHealthCheck
	elasticloadbalancing:CreateAppCookieStickinessPolicy
	elasticloadbalancing:CreateLBCookieStickinessPolicy
	elasticloadbalancing:CreateListener
	elasticloadbalancing:CreateLoadBalancer
	elasticloadbalancing:CreateLoadBalancerListeners
	elasticloadbalancing:CreateLoadBalancerPolicy
	elasticloadbalancing:CreateRule
	elasticloadbalancing:CreateTargetGroup
	elasticloadbalancing:CreateTrustStore
	elasticloadbalancing:DeleteListener
	elasticloadbalancing:DeleteLoadBalancer
	elasticloadbalancing:DeleteLoadBalancerListeners
	elasticloadbalancing:DeleteLoadBalancerPolicy
	elasticloadbalancing:DeleteRule
	elasticloadbalancing:DeleteSharedTrustStoreAssociation
	elasticloadbalancing:DeleteTargetGroup

服务前缀	操作
	elasticloadbalancing:DeleteTrustStore
	elasticloadbalancing:DeregisterInstancesFromLoadBalancer
	elasticloadbalancing:DeregisterTargets
	elasticloadbalancing:DescribeAccountLimits
	elasticloadbalancing:DescribeCapacityReservation
	elasticloadbalancing:DescribeInstanceHealth
	elasticloadbalancing:DescribeListenerAttributes
	elasticloadbalancing:DescribeListenerCertificates
	elasticloadbalancing:DescribeListeners
	elasticloadbalancing:DescribeLoadBalancerAttributes
	elasticloadbalancing:DescribeLoadBalancerPolicies
	elasticloadbalancing:DescribeLoadBalancerPolicyTypes
	elasticloadbalancing:DescribeLoadBalancers
	elasticloadbalancing:DescribeRules
	elasticloadbalancing:DescribeSSLPolicies
	elasticloadbalancing:DescribeTargetGroupAttributes
	elasticloadbalancing:DescribeTargetGroups
	elasticloadbalancing:DescribeTargetHealth
	elasticloadbalancing:DescribeTrustStoreAssociations
	elasticloadbalancing:DescribeTrustStoreRevocations
	elasticloadbalancing:DescribeTrustStores

服务前缀	操作
	elasticloadbalancing:DetachLoadBalancerFromSubnets
	elasticloadbalancing:DisableAvailabilityZonesForLoadBalancer
	elasticloadbalancing:EnableAvailabilityZonesForLoadBalancer
	elasticloadbalancing:GetResourcePolicy
	elasticloadbalancing:GetTrustStoreCaCertificatesBundle
	elasticloadbalancing:GetTrustStoreRevocationContent
	elasticloadbalancing:ModifyCapacityReservation
	elasticloadbalancing:ModifyIpPools
	elasticloadbalancing:ModifyListener
	elasticloadbalancing:ModifyLoadBalancerAttributes
	elasticloadbalancing:ModifyRule
	elasticloadbalancing:ModifyTargetGroup
	elasticloadbalancing:ModifyTargetGroupAttributes
	elasticloadbalancing:ModifyTrustStore
	elasticloadbalancing:RegisterInstancesWithLoadBalancer
	elasticloadbalancing:RegisterTargets
	elasticloadbalancing:RemoveListenerCertificates
	elasticloadbalancing:RemoveTrustStoreRevocations
	elasticloadbalancing:SetIpAddressType
	elasticloadbalancing:SetLoadBalancerListenerSSLCertificate
	elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer

服务前缀	操作
	elasticloadbalancing:SetLoadBalancerPoliciesOfListener elasticloadbalancing:SetRulePriorities elasticloadbalancing:SetSecurityGroups elasticloadbalancing:SetSubnets
elastictranscoder	elastictranscoder:CancelJob elastictranscoder:CreateJob elastictranscoder:CreatePipeline elastictranscoder:CreatePreset elastictranscoder:DeletePipeline elastictranscoder:DeletePreset elastictranscoder:ListJobsByPipeline elastictranscoder:ListJobsByStatus elastictranscoder:ListPipelines elastictranscoder:ListPresets elastictranscoder:ReadJob elastictranscoder:ReadPipeline elastictranscoder:ReadPreset elastictranscoder:TestRole elastictranscoder:UpdatePipeline elastictranscoder:UpdatePipelineNotifications elastictranscoder:UpdatePipelineStatus

服务前缀	操作
emr-containers	emr-containers:CancelJobRun
	emr-containers:CreateJobTemplate
	emr-containers:CreateManagedEndpoint
	emr-containers:CreateSecurityConfiguration
	emr-containers:CreateVirtualCluster
	emr-containers:DeleteJobTemplate
	emr-containers:DeleteManagedEndpoint
	emr-containers:DeleteVirtualCluster
	emr-containers:DescribeJobRun
	emr-containers:DescribeJobTemplate
	emr-containers:DescribeManagedEndpoint
	emr-containers:DescribeSecurityConfiguration
	emr-containers:DescribeVirtualCluster
	emr-containers:GetManagedEndpointSessionCredentials
	emr-containers:ListJobRuns
	emr-containers:ListJobTemplates
	emr-containers:ListManagedEndpoints
	emr-containers:ListSecurityConfigurations
	emr-containers:ListVirtualClusters
	emr-containers:StartJobRun

服务前缀	操作
emr-serverless	emr-serverless:CancelJobRun emr-serverless:CreateApplication emr-serverless:DeleteApplication emr-serverless:GetApplication emr-serverless:GetDashboardForJobRun emr-serverless:GetJobRun emr-serverless:ListApplications emr-serverless:ListJobRunAttempts emr-serverless:ListJobRuns emr-serverless:StartApplication emr-serverless:StartJobRun emr-serverless:StopApplication emr-serverless:UpdateApplication

服务前缀	操作
es	es:AcceptInboundConnection
	es:AcceptInboundCrossClusterSearchConnection
	es:AssociatePackage
	es:AuthorizeVpcEndpointAccess
	es:CancelElasticsearchServiceSoftwareUpdate
	es:CancelServiceSoftwareUpdate
	es:CreateDomain
	es:CreateElasticsearchDomain
	es:CreateIndex
	es:CreateOutboundConnection
	es:CreateOutboundCrossClusterSearchConnection
	es:CreatePackage
	es:CreateVpcEndpoint
	es:DeleteDomain
	es:DeleteElasticsearchDomain
	es:DeleteElasticsearchServiceRole
	es:DeleteInboundConnection
	es:DeleteInboundCrossClusterSearchConnection
	es:DeleteIndex
	es:DeleteOutboundConnection
	es:DeleteOutboundCrossClusterSearchConnection

服务前缀	操作
	es:DeletePackage
	es:DeleteVpcEndpoint
	es:DescribeDomain
	es:DescribeDomainAutoTunes
	es:DescribeDomainChangeProgress
	es:DescribeDomainConfig
	es:DescribeDomainHealth
	es:DescribeDomainNodes
	es:DescribeDomains
	es:DescribeDryRunProgress
	es:DescribeElasticsearchDomain
	es:DescribeElasticsearchDomainConfig
	es:DescribeElasticsearchDomains
	es:DescribeElasticsearchInstanceTypeLimits
	es:DescribeInboundConnections
	es:DescribeInboundCrossClusterSearchConnections
	es:DescribeInstanceTypeLimits
	es:DescribeOutboundConnections
	es:DescribeOutboundCrossClusterSearchConnections
	es:DescribePackages
	es:DescribeReservedElasticsearchInstanceOfferings

服务前缀	操作
	es:DescribeReservedElasticsearchInstances
	es:DescribeReservedInstanceOfferings
	es:DescribeReservedInstances
	es:DescribeVpcEndpoints
	es:DissociatePackage
	es:DissociatePackages
	es:GetCompatibleElasticsearchVersions
	es:GetCompatibleVersions
	es:GetDataSource
	es:GetDomainMaintenanceStatus
	es:GetPackageVersionHistory
	es:GetUpgradeHistory
	es:GetUpgradeStatus
	es:ListDataSources
	es:ListDomainNames
	es:ListDomainsForPackage
	es:ListElasticsearchInstanceTypes
	es:ListElasticsearchVersions
	es:ListInstanceTypeDetails
	es:ListPackagesForDomain
	es:ListScheduledActions

服务前缀	操作
	es:ListVersions
	es:ListVpcEndpointAccess
	es:ListVpcEndpoints
	es:ListVpcEndpointsForDomain
	es:PurchaseReservedElasticsearchInstanceOffering
	es:PurchaseReservedInstanceOffering
	es:RejectInboundConnection
	es:RejectInboundCrossClusterSearchConnection
	es:RevokeVpcEndpointAccess
	es:StartDomainMaintenance
	es:StartElasticsearchServiceSoftwareUpdate
	es:StartServiceSoftwareUpdate
	es:UpdateDataSource
	es:UpdateDomainConfig
	es:UpdateElasticsearchDomainConfig
	es:UpdateIndex
	es:UpdatePackage
	es:UpdatePackageScope
	es:UpdateScheduledAction
	es:UpdateVpcEndpoint
	es:UpgradeDomain

服务前缀	操作
	es:UpgradeElasticsearchDomain

服务前缀	操作
events	events:ActivateEventSource events:CancelReplay events:CreateApiDestination events:CreateArchive events:CreateConnection events:CreateEndpoint events:CreateEventBus events:CreatePartnerEventSource events:DeactivateEventSource events:DeauthorizeConnection events:DeleteApiDestination events:DeleteArchive events:DeleteConnection events:DeleteEndpoint events:DeleteEventBus events:DeletePartnerEventSource events:DeleteRule events:DescribeApiDestination events:DescribeArchive events:DescribeConnection events:DescribeEndpoint

服务前缀	操作
	events:DescribeEventBus events:DescribeEventSource events:DescribePartnerEventSource events:DescribeReplay events:DescribeRule events:DisableRule events:EnableRule events:ListApiDestinations events:ListArchives events:ListConnections events:ListEndpoints events:ListEventBuses events:ListEventSources events:ListPartnerEventSourceAccounts events:ListPartnerEventSources events:ListReplays events:ListRuleNamesByTarget events:ListRules events:ListTargetsByRule events:PutPermission events:PutRule

服务前缀	操作
	events:PutTargets events:RemovePermission events:RemoveTargets events:StartReplay events:TestEventPattern events:UpdateApiDestination events:UpdateArchive events:UpdateConnection events:UpdateEndpoint events:UpdateEventBus

服务前缀	操作
evidently	evidently:CreateExperiment
	evidently:CreateFeature
	evidently:CreateLaunch
	evidently:CreateProject
	evidently:CreateSegment
	evidently:DeleteExperiment
	evidently:DeleteFeature
	evidently:DeleteLaunch
	evidently:DeleteProject
	evidently:DeleteSegment
	evidently:GetExperiment
	evidently:GetExperimentResults
	evidently:GetFeature
	evidently:GetLaunch
	evidently:GetProject
	evidently:GetSegment
	evidently:ListExperiments
	evidently:ListFeatures
	evidently:ListLaunches
	evidently:ListProjects
	evidently:ListSegmentReferences

服务前缀	操作
	evidently:ListSegments
	evidently:StartExperiment
	evidently:StartLaunch
	evidently:StopExperiment
	evidently:StopLaunch
	evidently:TestSegmentPattern
	evidently:UpdateExperiment
	evidently:UpdateFeature
	evidently:UpdateLaunch
	evidently:UpdateProject
	evidently:UpdateProjectDataDelivery

服务前缀	操作
finspace	finspace:CreateEnvironment finspace:CreateKxChangeset finspace:CreateKxCluster finspace:CreateKxDatabase finspace:CreateKxDataview finspace:CreateKxEnvironment finspace:CreateKxScalingGroup finspace:CreateKxUser finspace:CreateKxVolume finspace:CreateUser finspace:DeleteEnvironment finspace:DeleteKxCluster finspace:DeleteKxClusterNode finspace:DeleteKxDatabase finspace:DeleteKxDataview finspace:DeleteKxEnvironment finspace:DeleteKxScalingGroup finspace:DeleteKxUser finspace:DeleteKxVolume finspace:GetEnvironment finspace:GetKxChangeset

服务前缀	操作
	finspace:GetKxCluster finspace:GetKxConnectionString finspace:GetKxDatabase finspace:GetKxDataview finspace:GetKxEnvironment finspace:GetKxScalingGroup finspace:GetKxUser finspace:GetKxVolume finspace:GetLoadSampleDataSetGroupIntoEnvironmentStatus finspace:GetUser finspace:ListEnvironments finspace:ListKxChangesets finspace:ListKxClusterNodes finspace:ListKxClusters finspace:ListKxDatabases finspace:ListKxDataviews finspace:ListKxEnvironments finspace:ListKxScalingGroups finspace:ListKxUsers finspace:ListKxVolumes finspace:ListUsers

服务前缀	操作
	finspace:LoadSampleDataSetGroupIntoEnvironment finspace:ResetUserPassword finspace:UpdateEnvironment finspace:UpdateKxClusterCodeConfiguration finspace:UpdateKxClusterDatabases finspace:UpdateKxDatabase finspace:UpdateKxDataview finspace:UpdateKxEnvironment finspace:UpdateKxEnvironmentNetwork finspace:UpdateKxUser finspace:UpdateKxVolume finspace:UpdateUser
Firehose	firehose:CreateDeliveryStream firehose:DeleteDeliveryStream firehose:DescribeDeliveryStream firehose:ListDeliveryStreams firehose:StartDeliveryStreamEncryption firehose:StopDeliveryStreamEncryption firehose:UpdateDestination

服务前缀	操作
fis	fis:CreateExperimentTemplate fis:CreateTargetAccountConfiguration fis:DeleteExperimentTemplate fis:DeleteTargetAccountConfiguration fis:GetAction fis:GetExperiment fis:GetExperimentTargetAccountConfiguration fis:GetExperimentTemplate fis:GetSafetyLever fis:GetTargetAccountConfiguration fis:GetTargetResourceType fis:ListActions fis:ListExperimentResolvedTargets fis:ListExperimentTargetAccountConfigurations fis:ListExperimentTemplates fis:ListExperiments fis:ListTargetAccountConfigurations fis:ListTargetResourceTypes fis:StartExperiment fis:StopExperiment fis:UpdateExperimentTemplate

服务前缀	操作
	fis:UpdateSafetyLeverState fis:UpdateTargetAccountConfiguration

服务前缀	操作
fms	fms:AssociateAdminAccount
	fms:AssociateThirdPartyFirewall
	fms:BatchAssociateResource
	fms:BatchDisassociateResource
	fms:DeleteAppsList
	fms:DeleteNotificationChannel
	fms:DeletePolicy
	fms:DeleteProtocolsList
	fms:DeleteResourceSet
	fms:DisassociateAdminAccount
	fms:DisassociateThirdPartyFirewall
	fms:GetAdminAccount
	fms:GetAdminScope
	fms:GetAppsList
	fms:GetComplianceDetail
	fms:GetNotificationChannel
	fms:GetPolicy
	fms:GetProtectionStatus
	fms:GetProtocolsList
	fms:GetResourceSet
	fms:GetThirdPartyFirewallAssociationStatus

服务前缀	操作
	fms:GetViolationDetails
	fms:ListAdminAccountsForOrganization
	fms:ListAdminsManagingAccount
	fms:ListAppsLists
	fms:ListComplianceStatus
	fms:ListDiscoveredResources
	fms:ListMemberAccounts
	fms:ListPolicies
	fms:ListProtocolsLists
	fms:ListResourceSetResources
	fms:ListResourceSets
	fms:ListThirdPartyFirewallFirewallPolicies
	fms:PutAdminAccount
	fms:PutAppsList
	fms:PutNotificationChannel
	fms:PutPolicy
	fms:PutProtocolsList
	fms:PutResourceSet

服务前缀	操作
frauddetector	frauddetector:BatchCreateVariable frauddetector:BatchGetVariable frauddetector:CancelBatchImportJob frauddetector:CancelBatchPredictionJob frauddetector:CreateBatchImportJob frauddetector:CreateBatchPredictionJob frauddetector:CreateDetectorVersion frauddetector:CreateList frauddetector:CreateModel frauddetector:CreateModelVersion frauddetector:CreateRule frauddetector:CreateVariable frauddetector:DeleteBatchImportJob frauddetector:DeleteBatchPredictionJob frauddetector:DeleteDetector frauddetector:DeleteDetectorVersion frauddetector:DeleteEntityType frauddetector:DeleteEvent frauddetector:DeleteEventType frauddetector:DeleteEventsByEventType frauddetector:DeleteExternalModel

服务前缀	操作
	frauddetector:DeleteLabel frauddetector:DeleteList frauddetector:DeleteModel frauddetector:DeleteModelVersion frauddetector:DeleteOutcome frauddetector:DeleteRule frauddetector:DeleteVariable frauddetector:DescribeDetector frauddetector:DescribeModelVersions frauddetector:GetBatchImportJobs frauddetector:GetBatchPredictionJobs frauddetector:GetDeleteEventsByEventTypeStatus frauddetector:GetDetectorVersion frauddetector:GetDetectors frauddetector:GetEntityTypes frauddetector:GetEvent frauddetector:GetEventPrediction frauddetector:GetEventPredictionMetadata frauddetector:GetEventTypes frauddetector:GetExternalModels frauddetector:GetKMSEncryptionKey

服务前缀	操作
	frauddetector:GetLabels frauddetector:GetListElements frauddetector:GetListsMetadata frauddetector:GetModelVersion frauddetector:GetModels frauddetector:GetOutcomes frauddetector:GetRules frauddetector:GetVariables frauddetector:ListEventPredictions frauddetector:PutDetector frauddetector:PutEntityType frauddetector:PutEventType frauddetector:PutExternalModel frauddetector:PutKMSEncryptionKey frauddetector:PutLabel frauddetector:PutOutcome frauddetector:SendEvent frauddetector:UpdateDetectorVersion frauddetector:UpdateDetectorVersionMetadata frauddetector:UpdateDetectorVersionStatus frauddetector:UpdateEventLabel

服务前缀	操作
	frauddetector:UpdateList frauddetector:UpdateModel frauddetector:UpdateModelVersion frauddetector:UpdateModelVersionStatus frauddetector:UpdateRuleMetadata frauddetector:UpdateRuleVersion frauddetector:UpdateVariable

服务前缀	操作
fsx	fsx:AssociateFileSystemAliases
	fsx:CancelDataRepositoryTask
	fsx:CopyBackup
	fsx:CreateDataRepositoryTask
	fsx:CreateFileCache
	fsx:CreateFileSystem
	fsx:CreateFileSystemFromBackup
	fsx:CreateSnapshot
	fsx:CreateStorageVirtualMachine
	fsx:CreateVolume
	fsx:CreateVolumeFromBackup
	fsx:DeleteBackup
	fsx:DeleteFileCache
	fsx:DeleteFileSystem
	fsx:DeleteSnapshot
	fsx:DeleteStorageVirtualMachine
	fsx:DeleteVolume
	fsx:DescribeBackups
	fsx:DescribeDataRepositoryAssociations
	fsx:DescribeDataRepositoryTasks
	fsx:DescribeFileCaches

服务前缀	操作
	fsx:DescribeFileSystemAliases
	fsx:DescribeFileSystems
	fsx:DescribeS3AccessPointAttachments
	fsx:DescribeSharedVpcConfiguration
	fsx:DescribeSnapshots
	fsx:DescribeStorageVirtualMachines
	fsx:DescribeVolumes
	fsx:DetachAndDeleteS3AccessPoint
	fsx:DisassociateFileSystemAliases
	fsx:ReleaseFileSystemNfsV3Locks
	fsx:RestoreVolumeFromSnapshot
	fsx:StartMisconfiguredStateRecovery
	fsx:UpdateDataRepositoryAssociation
	fsx:UpdateFileCache
	fsx:UpdateFileSystem
	fsx:UpdateSharedVpcConfiguration
	fsx:UpdateSnapshot
	fsx:UpdateStorageVirtualMachine
	fsx:UpdateVolume

服务前缀	操作
gamelift	gamelift:AcceptMatch gamelift:ClaimGameServer gamelift:CreateAlias gamelift:CreateBuild gamelift:CreateContainerGroupDefinition gamelift:CreateFleet gamelift:CreateFleetLocations gamelift:CreateGameServerGroup gamelift:CreateGameSession gamelift:CreateGameSessionQueue gamelift:CreateLocation gamelift:CreateMatchmakingConfiguration gamelift:CreateMatchmakingRuleSet gamelift:CreatePlayerSession gamelift:CreatePlayerSessions gamelift:CreateScript gamelift:CreateVpcPeeringAuthorization gamelift:CreateVpcPeeringConnection gamelift:DeleteAlias gamelift:DeleteBuild gamelift:DeleteContainerGroupDefinition

服务前缀	操作
	gamelift:DeleteFleet gamelift:DeleteFleetLocations gamelift:DeleteGameServerGroup gamelift:DeleteGameSessionQueue gamelift:DeleteLocation gamelift:DeleteMatchmakingConfiguration gamelift:DeleteMatchmakingRuleSet gamelift:DeleteScalingPolicy gamelift:DeleteScript gamelift:DeleteVpcPeeringAuthorization gamelift:DeleteVpcPeeringConnection gamelift:DeregisterCompute gamelift:DeregisterGameServer gamelift:DescribeAlias gamelift:DescribeBuild gamelift:DescribeCompute gamelift:DescribeContainerFleet gamelift:DescribeContainerGroupDefinition gamelift:DescribeEC2InstanceLimits gamelift:DescribeFleetAttributes gamelift:DescribeFleetCapacity

服务前缀	操作
	gamelift:DescribeFleetEvents gamelift:DescribeFleetLocationAttributes gamelift:DescribeFleetLocationCapacity gamelift:DescribeFleetLocationUtilization gamelift:DescribeFleetPortSettings gamelift:DescribeFleetUtilization gamelift:DescribeGameServer gamelift:DescribeGameServerGroup gamelift:DescribeGameServerInstances gamelift:DescribeGameSessionDetails gamelift:DescribeGameSessionPlacement gamelift:DescribeGameSessionQueues gamelift:DescribeGameSessions gamelift:DescribeInstances gamelift:DescribeMatchmaking gamelift:DescribeMatchmakingConfigurations gamelift:DescribeMatchmakingRuleSets gamelift:DescribePlayerSessions gamelift:DescribeRuntimeConfiguration gamelift:DescribeScalingPolicies gamelift:DescribeScript

服务前缀	操作
	gamelift:DescribeVpcPeeringAuthorizations gamelift:DescribeVpcPeeringConnections gamelift:GetComputeAccess gamelift:GetComputeAuthToken gamelift:GetGameSessionLogUrl gamelift:GetInstanceAccess gamelift:ListAliases gamelift:ListBuilds gamelift:ListCompute gamelift:ListContainerFleets gamelift:ListContainerGroupDefinitionVersions gamelift:ListContainerGroupDefinitions gamelift:ListFleetDeployments gamelift:ListFleets gamelift:ListGameServerGroups gamelift:ListGameServers gamelift:ListLocations gamelift:ListScripts gamelift:PutScalingPolicy gamelift:RegisterCompute gamelift:RegisterGameServer

服务前缀	操作
	gamelift:RequestUploadCredentials gamelift:ResolveAlias gamelift:ResumeGameServerGroup gamelift:SearchGameSessions gamelift:StartFleetActions gamelift:StartGameSessionPlacement gamelift:StartMatchBackfill gamelift:StartMatchmaking gamelift:StopFleetActions gamelift:StopGameSessionPlacement gamelift:StopMatchmaking gamelift:SuspendGameServerGroup gamelift:TerminateGameSession gamelift:UpdateAlias gamelift:UpdateBuild gamelift:UpdateContainerGroupDefinition gamelift:UpdateFleetAttributes gamelift:UpdateFleetCapacity gamelift:UpdateFleetPortSettings gamelift:UpdateGameServer gamelift:UpdateGameServerGroup

服务前缀	操作
	gamelift:UpdateGameSession gamelift:UpdateGameSessionQueue gamelift:UpdateMatchmakingConfiguration gamelift:UpdateRuntimeConfiguration gamelift:UpdateScript gamelift:ValidateMatchmakingRuleSet

服务前缀	操作
geo	geo:AssociateTrackerConsumer
	geo:BatchDeleteDevicePositionHistory
	geo:BatchDeleteGeofence
	geo:BatchEvaluateGeofences
	geo:BatchGetDevicePosition
	geo:BatchPutGeofence
	geo:BatchUpdateDevicePosition
	geo:CalculateRoute
	geo:CalculateRouteMatrix
	geo:CreateGeofenceCollection
	geo:CreateMap
	geo:CreatePlaceIndex
	geo:CreateRouteCalculator
	geo:CreateTracker
	geo:DeleteGeofenceCollection
	geo:DeleteKey
	geo:DeleteMap
	geo:DeletePlaceIndex
	geo:DeleteRouteCalculator
	geo:DeleteTracker
	geo:DescribeGeofenceCollection

服务前缀	操作
	geo:DescribeKey
	geo:DescribeMap
	geo:DescribePlaceIndex
	geo:DescribeRouteCalculator
	geo:DescribeTracker
	geo:DisassociateTrackerConsumer
	geo:ForecastGeofenceEvents
	geo:GetDevicePosition
	geo:GetDevicePositionHistory
	geo:GetGeofence
	geo:GetMapGlyphs
	geo:GetMapSprites
	geo:GetMapStyleDescriptor
	geo:GetMapTile
	geo:GetPlace
	geo:ListDevicePositions
	geo:ListGeofenceCollections
	geo:ListGeofences
	geo:ListKeys
	geo:ListMaps
	geo:ListPlaceIndexes

服务前缀	操作
	geo:ListRouteCalculators
	geo:ListTrackerConsumers
	geo:ListTrackers
	geo:PutGeofence
	geo:SearchPlaceIndexForPosition
	geo:SearchPlaceIndexForSuggestions
	geo:SearchPlaceIndexForText
	geo:UpdateGeofenceCollection
	geo:UpdateKey
	geo:UpdateMap
	geo:UpdatePlaceIndex
	geo:UpdateRouteCalculator
	geo:UpdateTracker
	geo:VerifyDevicePosition

服务前缀	操作
glacier	glacier:AbortMultipartUpload glacier:AbortVaultLock glacier:CompleteMultipartUpload glacier:CompleteVaultLock glacier:CreateVault glacier:DeleteArchive glacier:DeleteVault glacier:DeleteVaultAccessPolicy glacier:DeleteVaultNotifications glacier:DescribeJob glacier:DescribeVault glacier:GetDataRetrievalPolicy glacier:GetJobOutput glacier:GetVaultAccessPolicy glacier:GetVaultLock glacier:GetVaultNotifications glacier:InitiateJob glacier:InitiateMultipartUpload glacier:InitiateVaultLock glacier:ListJobs glacier:ListMultipartUploads

服务前缀	操作
	glacier:ListParts glacier:ListProvisionedCapacity glacier:ListVaults glacier:PurchaseProvisionedCapacity glacier:SetDataRetrievalPolicy glacier:SetVaultAccessPolicy glacier:SetVaultNotifications glacier:UploadArchive glacier:UploadMultipartPart

服务前缀	操作
grafana	grafana:AssociateLicense
	grafana:CreateWorkspace
	grafana:CreateWorkspaceApiKey
	grafana:CreateWorkspaceServiceAccount
	grafana:CreateWorkspaceServiceAccountToken
	grafana:DeleteWorkspace
	grafana:DeleteWorkspaceApiKey
	grafana:DeleteWorkspaceServiceAccount
	grafana:DeleteWorkspaceServiceAccountToken
	grafana:DescribeWorkspace
	grafana:DescribeWorkspaceAuthentication
	grafana:DescribeWorkspaceConfiguration
	grafana:DisassociateLicense
	grafana:ListPermissions
	grafana:ListVersions
	grafana:ListWorkspaceServiceAccountTokens
	grafana:ListWorkspaceServiceAccounts
	grafana:ListWorkspaces
	grafana:UpdatePermissions
	grafana:UpdateWorkspace
	grafana:UpdateWorkspaceAuthentication

服务前缀	操作
	grafana:UpdateWorkspaceConfiguration

服务前缀	操作
greengrass	greengrass:AssociateRoleToGroup greengrass:AssociateServiceRoleToAccount greengrass:BatchAssociateClientDeviceWithCoreDevice greengrass:BatchDisassociateClientDeviceFromCoreDevice greengrass:CancelDeployment greengrass:CreateComponentVersion greengrass:CreateConnectorDefinition greengrass:CreateConnectorDefinitionVersion greengrass:CreateCoreDefinition greengrass:CreateCoreDefinitionVersion greengrass:CreateDeployment greengrass:CreateDeviceDefinition greengrass:CreateDeviceDefinitionVersion greengrass:CreateFunctionDefinition greengrass:CreateFunctionDefinitionVersion greengrass:CreateGroup greengrass:CreateGroupCertificateAuthority greengrass:CreateGroupVersion greengrass:CreateLoggerDefinition greengrass:CreateLoggerDefinitionVersion greengrass:CreateResourceDefinition

服务前缀	操作
	greengrass:CreateResourceDefinitionVersion greengrass:CreateSoftwareUpdateJob greengrass:CreateSubscriptionDefinition greengrass:CreateSubscriptionDefinitionVersion greengrass:DeleteComponent greengrass:DeleteConnectorDefinition greengrass:DeleteCoreDefinition greengrass:DeleteCoreDevice greengrass:DeleteDeployment greengrass:DeleteDeviceDefinition greengrass:DeleteFunctionDefinition greengrass:DeleteGroup greengrass:DeleteLoggerDefinition greengrass:DeleteResourceDefinition greengrass:DeleteSubscriptionDefinition greengrass:DescribeComponent greengrass:DisassociateRoleFromGroup greengrass:DisassociateServiceRoleFromAccount greengrass:GetAssociatedRole greengrass:GetBulkDeploymentStatus greengrass:GetComponent

服务前缀	操作
	greengrass:GetComponentVersionArtifact greengrass:GetConnectivityInfo greengrass:GetConnectorDefinition greengrass:GetConnectorDefinitionVersion greengrass:GetCoreDefinition greengrass:GetCoreDefinitionVersion greengrass:GetCoreDevice greengrass:GetDeployment greengrass:GetDeploymentStatus greengrass:GetDeviceDefinition greengrass:GetDeviceDefinitionVersion greengrass:GetFunctionDefinition greengrass:GetFunctionDefinitionVersion greengrass:GetGroup greengrass:GetGroupCertificateAuthority greengrass:GetGroupCertificateConfiguration greengrass:GetGroupVersion greengrass:GetLoggerDefinition greengrass:GetLoggerDefinitionVersion greengrass:GetResourceDefinition greengrass:GetResourceDefinitionVersion

服务前缀	操作
	greengrass:GetServiceRoleForAccount greengrass:GetSubscriptionDefinition greengrass:GetSubscriptionDefinitionVersion greengrass:GetThingRuntimeConfiguration greengrass:ListBulkDeploymentDetailedReports greengrass:ListBulkDeployments greengrass:ListClientDevicesAssociatedWithCoreDevice greengrass:ListComponentVersions greengrass:ListComponents greengrass:ListConnectorDefinitionVersions greengrass:ListConnectorDefinitions greengrass:ListCoreDefinitionVersions greengrass:ListCoreDefinitions greengrass:ListCoreDevices greengrass:ListDeployments greengrass:ListDeviceDefinitionVersions greengrass:ListDeviceDefinitions greengrass:ListEffectiveDeployments greengrass:ListFunctionDefinitionVersions greengrass:ListFunctionDefinitions greengrass:ListGroupCertificateAuthorities

服务前缀	操作
	greengrass:ListGroupVersions greengrass:ListGroups greengrass:ListInstalledComponents greengrass:ListLoggerDefinitionVersions greengrass:ListLoggerDefinitions greengrass:ListResourceDefinitionVersions greengrass:ListResourceDefinitions greengrass:ListSubscriptionDefinitionVersions greengrass:ListSubscriptionDefinitions greengrass:ResetDeployments greengrass:StartBulkDeployment greengrass:StopBulkDeployment greengrass:UpdateConnectivityInfo greengrass:UpdateConnectorDefinition greengrass:UpdateCoreDefinition greengrass:UpdateDeviceDefinition greengrass:UpdateFunctionDefinition greengrass:UpdateGroup greengrass:UpdateGroupCertificateConfiguration greengrass:UpdateLoggerDefinition greengrass:UpdateResourceDefinition

服务前缀	操作
	greengrass:UpdateSubscriptionDefinition greengrass:UpdateThingRuntimeConfiguration

服务前缀	操作
groundstation	groundstation:CancelContact
	groundstation:CreateConfig
	groundstation:CreateDataflowEndpointGroup
	groundstation:CreateDataflowEndpointGroupV2
	groundstation:CreateEphemeris
	groundstation:CreateMissionProfile
	groundstation:DeleteConfig
	groundstation:DeleteDataflowEndpointGroup
	groundstation:DeleteEphemeris
	groundstation:DeleteMissionProfile
	groundstation:DescribeContact
	groundstation:DescribeEphemeris
	groundstation:GetConfig
	groundstation:GetDataflowEndpointGroup
	groundstation:GetMinuteUsage
	groundstation:GetMissionProfile
	groundstation:GetSatellite
	groundstation:ListConfigs
	groundstation:ListContacts
	groundstation:ListDataflowEndpointGroups
	groundstation:ListEphemerides

服务前缀	操作
	groundstation:ListGroundStations
	groundstation:ListMissionProfiles
	groundstation:ListSatellites
	groundstation:RegisterAgent
	groundstation:ReserveContact
	groundstation:UpdateAgentStatus
	groundstation:UpdateConfig
	groundstation:UpdateEphemeris
	groundstation:UpdateMissionProfile

服务前缀	操作
guardduty	guardduty:AcceptAdministratorInvitation guardduty:AcceptInvitation guardduty:ArchiveFindings guardduty:CreateDetector guardduty:CreateFilter guardduty:CreateIPSet guardduty:CreateMalwareProtectionPlan guardduty:CreateMembers guardduty:CreatePublishingDestination guardduty:CreateSampleFindings guardduty:CreateThreatEntitySet guardduty:CreateThreatIntelSet guardduty:CreateTrustedEntitySet guardduty:DeclineInvitations guardduty:DeleteDetector guardduty:DeleteFilter guardduty:DeleteIPSet guardduty:DeleteInvitations guardduty:DeleteMalwareProtectionPlan guardduty:DeleteMembers guardduty:DeletePublishingDestination

服务前缀	操作
	guardduty:DeleteThreatEntitySet guardduty:DeleteThreatIntelSet guardduty:DeleteTrustedEntitySet guardduty:DescribeMalwareScans guardduty:DescribeOrganizationConfiguration guardduty:DescribePublishingDestination guardduty:DisableOrganizationAdminAccount guardduty:DisassociateFromAdministratorAccount guardduty:DisassociateFromMasterAccount guardduty:DisassociateMembers guardduty:EnableOrganizationAdminAccount guardduty:GetAdministratorAccount guardduty:GetCoverageStatistics guardduty:GetDetector guardduty:GetFilter guardduty:GetFindings guardduty:GetFindingsStatistics guardduty:GetIPSet guardduty:GetInvitationsCount guardduty:GetMalwareProtectionPlan guardduty:GetMalwareScan

服务前缀	操作
	guardduty:GetMalwareScanSettings guardduty:GetMasterAccount guardduty:GetMemberDetectors guardduty:GetMembers guardduty:GetOrganizationStatistics guardduty:GetRemainingFreeTrialDays guardduty:GetThreatEntitySet guardduty:GetThreatIntelSet guardduty:GetTrustedEntitySet guardduty:GetUsageStatistics guardduty:InviteMembers guardduty:ListCoverage guardduty:ListDetectors guardduty:ListFilters guardduty:ListFindings guardduty:ListIPSets guardduty:ListInvitations guardduty:ListMalwareProtectionPlans guardduty:ListMalwareScans guardduty:ListMembers guardduty:ListOrganizationAdminAccounts

服务前缀	操作
	guardduty:ListPublishingDestinations guardduty:ListThreatEntitySets guardduty:ListThreatIntelSets guardduty:ListTrustedEntitySets guardduty:StartMalwareScan guardduty:StartMonitoringMembers guardduty:StopMonitoringMembers guardduty:UnarchiveFindings guardduty:UpdateDetector guardduty:UpdateFilter guardduty:UpdateFindingsFeedback guardduty:UpdateIPSet guardduty:UpdateMalwareProtectionPlan guardduty:UpdateMalwareScanSettings guardduty:UpdateMemberDetectors guardduty:UpdateOrganizationConfiguration guardduty:UpdatePublishingDestination guardduty:UpdateThreatEntitySet guardduty:UpdateThreatIntelSet guardduty:UpdateTrustedEntitySet

服务前缀	操作
healthlake	healthlake:CancelFHIRExportJobWithDelete
	healthlake:CreateFHIRDatastore
	healthlake:CreateResource
	healthlake:DeleteFHIRDatastore
	healthlake:DeleteResource
	healthlake:DescribeFHIRDatastore
	healthlake:DescribeFHIRExportJob
	healthlake:DescribeFHIRExportJobWithGet
	healthlake:DescribeFHIRImportJob
	healthlake:GetCapabilities
	healthlake:ListFHIRDatastores
	healthlake:ListFHIRExportJobs
	healthlake:ListFHIRImportJobs
	healthlake:ReadResource
	healthlake:SearchEverything
	healthlake:SearchWithGet
	healthlake:SearchWithPost
	healthlake:StartFHIRExportJob
	healthlake:StartFHIRExportJobWithPost
	healthlake:StartFHIRImportJob
	healthlake:UpdateResource

服务前缀	操作
honeycode	honeycode:BatchCreateTableRows honeycode:BatchDeleteTableRows honeycode:BatchUpdateTableRows honeycode:BatchUpsertTableRows honeycode:DescribeTableDataImportJob honeycode:GetScreenData honeycode:InvokeScreenAutomation honeycode:ListTableColumns honeycode:ListTableRows honeycode:ListTables honeycode:QueryTableRows honeycode:StartTableDataImportJob

服务前缀	操作
IAM	iam:AddClientIDToOpenIDConnectProvider
	iam:AddRoleToInstanceProfile
	iam:AddUserToGroup
	iam:AttachGroupPolicy
	iam:AttachRolePolicy
	iam:AttachUserPolicy
	iam:ChangePassword
	iam:CreateAccessKey
	iam:CreateAccountAlias
	iam:CreateGroup
	iam:CreateInstanceProfile
	iam:CreateLoginProfile
	iam:CreateOpenIDConnectProvider
	iam:CreatePolicy
	iam:CreatePolicyVersion
	iam:CreateRole
	iam:CreateSAMLProvider
	iam:CreateServiceLinkedRole
	iam:CreateServiceSpecificCredential
	iam:CreateUser
	iam:CreateVirtualMFADevice

服务前缀	操作
	iam:DeactivateMFADevice
	iam:DeleteAccessKey
	iam:DeleteAccountAlias
	iam:DeleteAccountPasswordPolicy
	iam:DeleteCloudFrontPublicKey
	iam:DeleteGroup
	iam:DeleteGroupPolicy
	iam:DeleteInstanceProfile
	iam:DeleteLoginProfile
	iam:DeleteOpenIDConnectProvider
	iam:DeletePolicy
	iam:DeletePolicyVersion
	iam:DeleteRole
	iam:DeleteRolePermissionsBoundary
	iam:DeleteRolePolicy
	iam:DeleteSAMLProvider
	iam:DeleteSSHPublicKey
	iam:DeleteServerCertificate
	iam:DeleteServiceLinkedRole
	iam:DeleteServiceSpecificCredential
	iam:DeleteSigningCertificate

服务前缀	操作
	iam:DeleteUser
	iam:DeleteUserPermissionsBoundary
	iam:DeleteUserPolicy
	iam:DeleteVirtualMFADevice
	iam:DetachGroupPolicy
	iam:DetachRolePolicy
	iam:DetachUserPolicy
	iam:DisableOrganizationsRootCredentialsManagement
	iam:DisableOrganizationsRootSessions
	iam:DisableOutboundWebIdentityFederation
	iam:EnableMFADevice
	iam:EnableOrganizationsRootCredentialsManagement
	iam:EnableOrganizationsRootSessions
	iam:EnableOutboundWebIdentityFederation
	iam:GenerateCredentialReport
	iam:GenerateOrganizationsAccessReport
	iam:GenerateServiceLastAccessedDetails
	iam:GetAccessKeyLastUsed
	iam:GetAccountAuthorizationDetails
	iam:GetAccountEmailAddress
	iam:GetAccountName

服务前缀	操作
	iam:GetAccountPasswordPolicy
	iam:GetAccountSummary
	iam:GetCloudFrontPublicKey
	iam:GetContextKeysForCustomPolicy
	iam:GetContextKeysForPrincipalPolicy
	iam:GetCredentialReport
	iam:GetGroup
	iam:GetGroupPolicy
	iam:GetInstanceProfile
	iam:GetLoginProfile
	iam:GetMFADevice
	iam:GetOpenIDConnectProvider
	iam:GetOrganizationsAccessReport
	iam:GetOutboundWebIdentityFederationInfo
	iam:GetPolicy
	iam:GetPolicyVersion
	iam:GetRole
	iam:GetRolePolicy
	iam:GetSAMLProvider
	iam:GetSSHPublicKey
	iam:GetServerCertificate

服务前缀	操作
	iam:GetServiceLastAccessedDetails
	iam:GetServiceLastAccessedDetailsWithEntities
	iam:GetServiceLinkedRoleDeletionStatus
	iam:GetUser
	iam:GetUserPolicy
	iam:ListAccessKeys
	iam:ListAccountAliases
	iam:ListAttachedGroupPolicies
	iam:ListAttachedRolePolicies
	iam:ListAttachedUserPolicies
	iam:ListCloudFrontPublicKeys
	iam:ListDelegationRequests
	iam:ListEntitiesForPolicy
	iam:ListGroupPolicies
	iam:ListGroups
	iam:ListGroupsForUser
	iam:ListInstanceProfiles
	iam:ListInstanceProfilesForRole
	iam:ListMFADevices
	iam:ListOpenIDConnectProviders
	iam:ListOrganizationsFeatures

服务前缀	操作
	iam:ListPolicies
	iam:ListPoliciesGrantingServiceAccess
	iam:ListPolicyVersions
	iam:ListRolePolicies
	iam:ListRoles
	iam:ListSAMLProviders
	iam:ListSSHPublicKeys
	iam:ListSTSRegionalEndpointsStatus
	iam:ListServerCertificates
	iam:ListServiceSpecificCredentials
	iam:ListSigningCertificates
	iam:ListUserPolicies
	iam:ListUsers
	iam:ListVirtualMFADevices
	iam:PutGroupPolicy
	iam:PutRolePermissionsBoundary
	iam:PutRolePolicy
	iam:PutUserPermissionsBoundary
	iam:PutUserPolicy
	iam:RemoveClientIDFromOpenIDConnectProvider
	iam:RemoveRoleFromInstanceProfile

服务前缀	操作
	iam:RemoveUserFromGroup
	iam:ResetServiceSpecificCredential
	iam:ResyncMFADevice
	iam:SetDefaultPolicyVersion
	iam:SetSTSRegionalEndpointStatus
	iam:SetSecurityTokenServicePreferences
	iam:SimulateCustomPolicy
	iam:SimulatePrincipalPolicy
	iam:UpdateAccessKey
	iam:UpdateAccountEmailAddress
	iam:UpdateAccountName
	iam:UpdateAccountPasswordPolicy
	iam:UpdateAssumeRolePolicy
	iam:UpdateCloudFrontPublicKey
	iam:UpdateGroup
	iam:UpdateLoginProfile
	iam:UpdateOpenIDConnectProviderThumbprint
	iam:UpdateRole
	iam:UpdateRoleDescription
	iam:UpdateSAMLProvider
	iam:UpdateSSHPublicKey

服务前缀	操作
	iam:UpdateServerCertificate iam:UpdateServiceSpecificCredential iam:UpdateSigningCertificate iam:UpdateUser iam:UploadCloudFrontPublicKey iam:UploadSSHPublicKey iam:UploadServerCertificate iam:UploadSigningCertificate

服务前缀	操作
identitystore	identitystore:CreateGroup identitystore:CreateGroupMembership identitystore:CreateUser identitystore:DeleteGroup identitystore:DeleteGroupMembership identitystore:DeleteUser identitystore:DescribeGroup identitystore:DescribeGroupMembership identitystore:DescribeUser identitystore:GetGroupId identitystore:GetGroupMembershipId identitystore:GetUserId identitystore:IsMemberInGroups identitystore:ListGroupMemberships identitystore:ListGroupMembershipsForMember identitystore:ListGroups identitystore:ListUsers identitystore:UpdateGroup identitystore:UpdateUser

服务前缀	操作
imagebuilder	imagebuilder:CancelImageCreation
	imagebuilder:CancelLifecycleExecution
	imagebuilder:CreateComponent
	imagebuilder:CreateContainerRecipe
	imagebuilder:CreateDistributionConfiguration
	imagebuilder:CreateImage
	imagebuilder:CreateImagePipeline
	imagebuilder:CreateImageRecipe
	imagebuilder:CreateInfrastructureConfiguration
	imagebuilder:CreateLifecyclePolicy
	imagebuilder:CreateWorkflow
	imagebuilder:DeleteComponent
	imagebuilder:DeleteContainerRecipe
	imagebuilder:DeleteDistributionConfiguration
	imagebuilder:DeleteImage
	imagebuilder:DeleteImagePipeline
	imagebuilder:DeleteImageRecipe
	imagebuilder:DeleteInfrastructureConfiguration
	imagebuilder:DeleteLifecyclePolicy
	imagebuilder:DeleteWorkflow
	imagebuilder:DistributeImage

服务前缀	操作
	imagebuilder:GetComponentPolicy
	imagebuilder:GetContainerRecipePolicy
	imagebuilder:GetImagePolicy
	imagebuilder:GetImageRecipePolicy
	imagebuilder:GetLifecycleExecution
	imagebuilder:GetLifecyclePolicy
	imagebuilder:GetMarketplaceResource
	imagebuilder:GetWorkflowExecution
	imagebuilder:GetWorkflowStepExecution
	imagebuilder:ImportComponent
	imagebuilder:ImportDiskImage
	imagebuilder:ImportVmImage
	imagebuilder:ListComponentBuildVersions
	imagebuilder:ListComponents
	imagebuilder:ListContainerRecipes
	imagebuilder:ListDistributionConfigurations
	imagebuilder:ListImageBuildVersions
	imagebuilder:ListImagePackages
	imagebuilder:ListImagePipelineImages
	imagebuilder:ListImagePipelines
	imagebuilder:ListImageRecipes

服务前缀	操作
	imagebuilder:ListImageScanFindingAggregations
	imagebuilder:ListImageScanFindings
	imagebuilder:ListImages
	imagebuilder:ListInfrastructureConfigurations
	imagebuilder:ListLifecycleExecutionResources
	imagebuilder:ListLifecycleExecutions
	imagebuilder:ListLifecyclePolicies
	imagebuilder:ListWaitingWorkflowSteps
	imagebuilder:ListWorkflowExecutions
	imagebuilder:ListWorkflowStepExecutions
	imagebuilder:ListWorkflows
	imagebuilder:PutComponentPolicy
	imagebuilder:PutContainerRecipePolicy
	imagebuilder:PutImagePolicy
	imagebuilder:PutImageRecipePolicy
	imagebuilder:RetryImage
	imagebuilder:SendWorkflowStepAction
	imagebuilder:StartImagePipelineExecution
	imagebuilder:StartResourceStateUpdate
	imagebuilder:UpdateDistributionConfiguration
	imagebuilder:UpdateImagePipeline

服务前缀	操作
	imagebuilder:UpdateInfrastructureConfiguration

服务前缀	操作
inspector	inspector:AddAttributesToFindings inspector:CreateAssessmentTarget inspector:CreateAssessmentTemplate inspector:CreateExclusionsPreview inspector:CreateResourceGroup inspector:DeleteAssessmentRun inspector:DeleteAssessmentTarget inspector:DeleteAssessmentTemplate inspector:DescribeAssessmentRuns inspector:DescribeAssessmentTargets inspector:DescribeAssessmentTemplates inspector:DescribeCrossAccountAccessRole inspector:DescribeExclusions inspector:DescribeFindings inspector:DescribeResourceGroups inspector:DescribeRulesPackages inspector:GetAssessmentReport inspector:GetExclusionsPreview inspector:GetTelemetryMetadata inspector:ListAssessmentRunAgents inspector:ListAssessmentRuns

服务前缀	操作
	inspector:ListAssessmentTargets inspector:ListAssessmentTemplates inspector:ListEventSubscriptions inspector:ListExclusions inspector:ListFindings inspector:ListRulesPackages inspector:PreviewAgents inspector:RegisterCrossAccountAccessRole inspector:RemoveAttributesFromFindings inspector:StartAssessmentRun inspector:StopAssessmentRun inspector:SubscribeToEvent inspector:UnsubscribeFromEvent inspector:UpdateAssessmentTarget

服务前缀	操作
inspector2	inspector2:AssociateMember
	inspector2:BatchGetAccountStatus
	inspector2:BatchGetCodeSnippet
	inspector2:BatchGetFindingDetails
	inspector2:BatchGetFreeTrialInfo
	inspector2:BatchGetMemberEc2DeepInspectionStatus
	inspector2:BatchUpdateMemberEc2DeepInspectionStatus
	inspector2:CancelFindingsReport
	inspector2:CancelSbomExport
	inspector2:CreateCisScanConfiguration
	inspector2:CreateCodeSecurityIntegration
	inspector2:CreateFilter
	inspector2:CreateFindingsReport
	inspector2:CreateSbomExport
	inspector2:DeleteCisScanConfiguration
	inspector2:DeleteCodeSecurityIntegration
	inspector2:DeleteFilter
	inspector2:DescribeOrganizationConfiguration
	inspector2:Disable
	inspector2:DisableDelegatedAdminAccount
	inspector2:DisassociateMember

服务前缀	操作
	inspector2:Enable
	inspector2:EnableDelegatedAdminAccount
	inspector2:GetCisScanReport
	inspector2:GetCisScanResultDetails
	inspector2:GetClustersForImage
	inspector2:GetCodeSecurityIntegration
	inspector2:GetCodeSecurityScan
	inspector2:GetConfiguration
	inspector2:GetDelegatedAdminAccount
	inspector2:GetEc2DeepInspectionConfiguration
	inspector2:GetEncryptionKey
	inspector2:GetFindingsReportStatus
	inspector2:GetMember
	inspector2:GetSbomExport
	inspector2:ListAccountPermissions
	inspector2:ListCisScanConfigurations
	inspector2:ListCisScanResultsAggregatedByChecks
	inspector2:ListCisScanResultsAggregatedByTargetResource
	inspector2:ListCisScans
	inspector2:ListCodeSecurityIntegrations
	inspector2:ListCodeSecurityScanConfigurations

服务前缀	操作
	inspector2:ListCoverage
	inspector2:ListCoverageStatistics
	inspector2:ListDelegatedAdminAccounts
	inspector2:ListFilters
	inspector2:ListFindingAggregations
	inspector2:ListFindings
	inspector2:ListMembers
	inspector2:ListUsageTotals
	inspector2:ResetEncryptionKey
	inspector2:SearchVulnerabilities
	inspector2:SendCisSessionHealth
	inspector2:SendCisSessionTelemetry
	inspector2:StartCisSession
	inspector2:StartCodeSecurityScan
	inspector2:StopCisSession
	inspector2:UpdateCisScanConfiguration
	inspector2:UpdateCodeSecurityIntegration
	inspector2:UpdateConfiguration
	inspector2:UpdateEc2DeepInspectionConfiguration
	inspector2:UpdateEncryptionKey
	inspector2:UpdateFilter

服务前缀	操作
	inspector2:UpdateOrgEc2DeepInspectionConfiguration inspector2:UpdateOrganizationConfiguration

服务前缀	操作
iot	iot:AcceptCertificateTransfer iot:AddThingToBillingGroup iot:AddThingToThingGroup iot:AssociateSbomWithPackageVersion iot:AssociateTargetsWithJob iot:AttachPolicy iot:AttachPrincipalPolicy iot:AttachSecurityProfile iot:AttachThingPrincipal iot:CancelAuditMitigationActionsTask iot:CancelAuditTask iot:CancelCertificateTransfer iot:CancelDetectMitigationActionsTask iot:CancelJob iot:CancelJobExecution iot:ClearDefaultAuthorizer iot:ConfirmTopicRuleDestination iot:CreateAuditSuppression iot:CreateAuthorizer iot:CreateBillingGroup iot:CreateCertificateFromCsr

服务前缀	操作
	iot:CreateCertificateProvider iot:CreateCommand iot:CreateCustomMetric iot:CreateDimension iot:CreateDomainConfiguration iot:CreateDynamicThingGroup iot:CreateFleetMetric iot:CreateJob iot:CreateJobTemplate iot:CreateKeysAndCertificate iot:CreateMitigationAction iot:CreateOTAUpdate iot:CreatePackage iot:CreatePackageVersion iot:CreatePolicy iot:CreatePolicyVersion iot:CreateProvisioningClaim iot:CreateProvisioningTemplate iot:CreateProvisioningTemplateVersion iot:CreateRoleAlias iot:CreateScheduledAudit

服务前缀	操作
	iot:CreateSecurityProfile iot:CreateStream iot:CreateThing iot:CreateThingGroup iot:CreateThingType iot:CreateTopicRule iot:CreateTopicRuleDestination iot:DeleteAccountAuditConfiguration iot:DeleteAuditSuppression iot:DeleteAuthorizer iot:DeleteBillingGroup iot:DeleteCACertificate iot:DeleteCertificate iot:DeleteCertificateProvider iot:DeleteCommand iot:DeleteCustomMetric iot:DeleteDimension iot:DeleteDomainConfiguration iot:DeleteDynamicThingGroup iot:DeleteFleetMetric iot:DeleteJob

服务前缀	操作
	iot:DeleteJobExecution iot:DeleteJobTemplate iot:DeleteMitigationAction iot:DeleteOTAUpdate iot:DeletePackage iot:DeletePackageVersion iot:DeletePolicy iot:DeletePolicyVersion iot:DeleteProvisioningTemplate iot:DeleteProvisioningTemplateVersion iot:DeleteRegistrationCode iot:DeleteRoleAlias iot:DeleteScheduledAudit iot:DeleteSecurityProfile iot:DeleteStream iot:DeleteThing iot:DeleteThingGroup iot:DeleteThingType iot:DeleteTopicRule iot:DeleteTopicRuleDestination iot:DeleteV2LoggingLevel

服务前缀	操作
	iot:DeprecateThingType iot:DescribeAccountAuditConfiguration iot:DescribeAuditFinding iot:DescribeAuditMitigationActionsTask iot:DescribeAuditSuppression iot:DescribeAuditTask iot:DescribeAuthorizer iot:DescribeBillingGroup iot:DescribeCACertificate iot:DescribeCertificate iot:DescribeCertificateProvider iot:DescribeCustomMetric iot:DescribeDefaultAuthorizer iot:DescribeDetectMitigationActionsTask iot:DescribeDimension iot:DescribeDomainConfiguration iot:DescribeEncryptionConfiguration iot:DescribeEndpoint iot:DescribeEventConfigurations iot:DescribeFleetMetric iot:DescribeIndex

服务前缀	操作
	iot:DescribeJob iot:DescribeJobExecution iot:DescribeJobTemplate iot:DescribeManagedJobTemplate iot:DescribeMitigationAction iot:DescribeProvisioningTemplate iot:DescribeProvisioningTemplateVersion iot:DescribeRoleAlias iot:DescribeScheduledAudit iot:DescribeSecurityProfile iot:DescribeStream iot:DescribeThing iot:DescribeThingGroup iot:DescribeThingRegistrationTask iot:DescribeThingType iot:DetachPolicy iot:DetachPrincipalPolicy iot:DetachSecurityProfile iot:DetachThingPrincipal iot:DisableTopicRule iot:DisassociateSbomFromPackageVersion

服务前缀	操作
	iot:EnableTopicRule iot:GetBehaviorModelTrainingSummaries iot:GetBucketsAggregation iot:GetCardinality iot:GetCommand iot:GetEffectivePolicies iot:GetJobDocument iot:GetLoggingOptions iot:GetOTAUpdate iot:GetPackage iot:GetPackageConfiguration iot:GetPackageVersion iot:GetPercentiles iot:GetPolicy iot:GetPolicyVersion iot:GetRegistrationCode iot:GetStatistics iot:GetThingConnectivityData iot:GetTopicRule iot:GetTopicRuleDestination iot:GetV2LoggingOptions

服务前缀	操作
	iot:ListActiveViolations iot:ListAttachedPolicies iot:ListAuditFindings iot:ListAuditMitigationActionsExecutions iot:ListAuditMitigationActionsTasks iot:ListAuditSuppressions iot:ListAuditTasks iot:ListAuthorizers iot:ListBillingGroups iot:ListCACertificates iot:ListCertificateProviders iot:ListCertificates iot:ListCertificatesByCA iot:ListCommands iot:ListCustomMetrics iot:ListDetectMitigationActionsExecutions iot:ListDetectMitigationActionsTasks iot:ListDimensions iot:ListDomainConfigurations iot:ListFleetMetrics iot:ListIndices

服务前缀	操作
	iot:ListJobExecutionsForJob iot:ListJobExecutionsForThing iot:ListJobTemplates iot:ListJobs iot:ListManagedJobTemplates iot:ListMetricValues iot:ListMitigationActions iot:ListOTAUpdates iot:ListOutgoingCertificates iot:ListPackageVersions iot:ListPackages iot:ListPolicies iot:ListPolicyPrincipals iot:ListPolicyVersions iot:ListPrincipalPolicies iot:ListPrincipalThings iot:ListPrincipalThingsV2 iot:ListProvisioningTemplateVersions iot:ListProvisioningTemplates iot:ListRelatedResourcesForAuditFinding iot:ListRoleAliases

服务前缀	操作
	iot:ListSbomValidationResults iot:ListScheduledAudits iot:ListSecurityProfiles iot:ListSecurityProfilesForTarget iot:ListStreams iot:ListTargetsForPolicy iot:ListTargetsForSecurityProfile iot:ListThingGroups iot:ListThingGroupsForThing iot:ListThingPrincipals iot:ListThingPrincipalsV2 iot:ListThingRegistrationTaskReports iot:ListThingRegistrationTasks iot:ListThingTypes iot:ListThings iot:ListThingsInBillingGroup iot:ListThingsInThingGroup iot:ListTopicRuleDestinations iot:ListTopicRules iot:ListV2LoggingLevels iot:ListViolationEvents

服务前缀	操作
	iot:PutVerificationStateOnViolation iot:RegisterCACertificate iot:RegisterCertificate iot:RegisterCertificateWithoutCA iot:RegisterThing iot:RejectCertificateTransfer iot:RemoveThingFromBillingGroup iot:RemoveThingFromThingGroup iot:ReplaceTopicRule iot:SearchIndex iot:SetDefaultAuthorizer iot:SetDefaultPolicyVersion iot:SetLoggingOptions iot:SetV2LoggingLevel iot:SetV2LoggingOptions iot:StartAuditMitigationActionsTask iot:StartDetectMitigationActionsTask iot:StartOnDemandAuditTask iot:StartThingRegistrationTask iot:StopThingRegistrationTask iot:TestAuthorization

服务前缀	操作
	iot:TestInvokeAuthorizer iot:TransferCertificate iot:UpdateAccountAuditConfiguration iot:UpdateAuditSuppression iot:UpdateAuthorizer iot:UpdateBillingGroup iot:UpdateCACertificate iot:UpdateCertificate iot:UpdateCertificateProvider iot:UpdateCommand iot:UpdateCustomMetric iot:UpdateDimension iot:UpdateDomainConfiguration iot:UpdateDynamicThingGroup iot:UpdateEncryptionConfiguration iot:UpdateEventConfigurations iot:UpdateFleetMetric iot:UpdateIndexingConfiguration iot:UpdateJob iot:UpdateMitigationAction iot:UpdatePackage

服务前缀	操作
	iot:UpdatePackageConfiguration iot:UpdatePackageVersion iot:UpdateProvisioningTemplate iot:UpdateRoleAlias iot:UpdateScheduledAudit iot:UpdateSecurityProfile iot:UpdateStream iot:UpdateThing iot:UpdateThingGroup iot:UpdateThingGroupsForThing iot:UpdateThingType iot:UpdateTopicRuleDestination iot:ValidateSecurityProfileBehaviors

服务前缀	操作
iotanalytics	iotanalytics:CancelPipelineReprocessing
	iotanalytics:CreateChannel
	iotanalytics:CreateDataset
	iotanalytics:CreateDatasetContent
	iotanalytics:CreateDatastore
	iotanalytics:CreatePipeline
	iotanalytics:DeleteChannel
	iotanalytics:DeleteDataset
	iotanalytics:DeleteDatasetContent
	iotanalytics:DeleteDatastore
	iotanalytics:DeletePipeline
	iotanalytics:DescribeChannel
	iotanalytics:DescribeDataset
	iotanalytics:DescribeDatastore
	iotanalytics:DescribeLoggingOptions
	iotanalytics:DescribePipeline
	iotanalytics:GetDatasetContent
	iotanalytics:ListChannels
	iotanalytics:ListDatasetContents
	iotanalytics:ListDatasets
	iotanalytics:ListDatastores

服务前缀	操作
	iotanalytics:ListPipelines iotanalytics:PutLoggingOptions iotanalytics:RunPipelineActivity iotanalytics:SampleChannelData iotanalytics:StartPipelineReprocessing iotanalytics:UpdateChannel iotanalytics:UpdateDataset iotanalytics:UpdateDatastore iotanalytics:UpdatePipeline
iotdeviceadvisor	iotdeviceadvisor:CreateSuiteDefinition iotdeviceadvisor:DeleteSuiteDefinition iotdeviceadvisor:GetEndpoint iotdeviceadvisor:GetSuiteDefinition iotdeviceadvisor:GetSuiteRun iotdeviceadvisor:GetSuiteRunReport iotdeviceadvisor:ListSuiteDefinitions iotdeviceadvisor:ListSuiteRuns iotdeviceadvisor:StartSuiteRun iotdeviceadvisor:StopSuiteRun iotdeviceadvisor:UpdateSuiteDefinition

服务前缀	操作
iotevents	iotevents:BatchAcknowledgeAlarm
	iotevents:BatchDeleteDetector
	iotevents:BatchDisableAlarm
	iotevents:BatchEnableAlarm
	iotevents:BatchResetAlarm
	iotevents:BatchSnoozeAlarm
	iotevents:BatchUpdateDetector
	iotevents:CreateAlarmModel
	iotevents:CreateDetectorModel
	iotevents:CreateInput
	iotevents:DeleteAlarmModel
	iotevents:DeleteDetectorModel
	iotevents:DeleteInput
	iotevents:DescribeAlarm
	iotevents:DescribeAlarmModel
	iotevents:DescribeDetector
	iotevents:DescribeDetectorModel
	iotevents:DescribeDetectorModelAnalysis
	iotevents:DescribeInput
	iotevents:DescribeLoggingOptions
	iotevents:GetDetectorModelAnalysisResults

服务前缀	操作
	iotevents:ListAlarmModelVersions iotevents:ListAlarmModels iotevents:ListAlarms iotevents:ListDetectorModelVersions iotevents:ListDetectorModels iotevents:ListDetectors iotevents:ListInputRoutings iotevents:ListInputs iotevents:PutLoggingOptions iotevents:StartDetectorModelAnalysis iotevents:UpdateAlarmModel iotevents:UpdateDetectorModel iotevents:UpdateInput
iotfleethub	iotfleethub:CreateApplication iotfleethub:DeleteApplication iotfleethub:DescribeApplication iotfleethub:ListApplications iotfleethub:UpdateApplication

服务前缀	操作
iotsitewise	iotsitewise:AssociateAssets
	iotsitewise:AssociateTimeSeriesToAssetProperty
	iotsitewise:BatchAssociateProjectAssets
	iotsitewise:BatchDisassociateProjectAssets
	iotsitewise:CreateAccessPolicy
	iotsitewise:CreateAsset
	iotsitewise:CreateAssetModel
	iotsitewise:CreateAssetModelCompositeModel
	iotsitewise:CreateBulkImportJob
	iotsitewise:CreateComputationModel
	iotsitewise:CreateDashboard
	iotsitewise:CreateDataset
	iotsitewise:CreateGateway
	iotsitewise:CreatePortal
	iotsitewise:CreateProject
	iotsitewise:DeleteAccessPolicy
	iotsitewise:DeleteAsset
	iotsitewise:DeleteAssetModel
	iotsitewise:DeleteAssetModelCompositeModel
	iotsitewise:DeleteComputationModel
	iotsitewise:DeleteDashboard

服务前缀	操作
	iotsitewise:DeleteDataset
	iotsitewise:DeleteGateway
	iotsitewise:DeletePortal
	iotsitewise:DeleteProject
	iotsitewise:DeleteTimeSeries
	iotsitewise:DescribeAccessPolicy
	iotsitewise:DescribeAsset
	iotsitewise:DescribeAssetCompositeModel
	iotsitewise:DescribeAssetModel
	iotsitewise:DescribeAssetModelCompositeModel
	iotsitewise:DescribeAssetModelInterfaceRelationship
	iotsitewise:DescribeAssetProperty
	iotsitewise:DescribeBulkImportJob
	iotsitewise:DescribeComputationModel
	iotsitewise:DescribeComputationModelExecutionSummary
	iotsitewise:DescribeDashboard
	iotsitewise:DescribeDataset
	iotsitewise:DescribeDefaultEncryptionConfiguration
	iotsitewise:DescribeExecution
	iotsitewise:DescribeGateway
	iotsitewise:DescribeGatewayCapabilityConfiguration

服务前缀	操作
	iotsitewise:DescribeLoggingOptions
	iotsitewise:DescribePortal
	iotsitewise:DescribeProject
	iotsitewise:DescribeStorageConfiguration
	iotsitewise:DescribeTimeSeries
	iotsitewise:DisassociateAssets
	iotsitewise:DisassociateTimeSeriesFromAssetProperty
	iotsitewise:ExecuteAction
	iotsitewise:ExecuteQuery
	iotsitewise:ListAccessPolicies
	iotsitewise:ListActions
	iotsitewise:ListAssetModelCompositeModels
	iotsitewise:ListAssetModelProperties
	iotsitewise:ListAssetModels
	iotsitewise:ListAssetProperties
	iotsitewise:ListAssetRelationships
	iotsitewise:ListAssets
	iotsitewise:ListAssociatedAssets
	iotsitewise:ListBulkImportJobs
	iotsitewise:ListCompositionRelationships
	iotsitewise:ListComputationModelDataBindingUsages

服务前缀	操作
	iotsitewise:ListComputationModelResolveToResources
	iotsitewise:ListComputationModels
	iotsitewise:ListDashboards
	iotsitewise:ListDatasets
	iotsitewise:ListExecutions
	iotsitewise:ListGateways
	iotsitewise:ListInterfaceRelationships
	iotsitewise:ListPortals
	iotsitewise:ListProjectAssets
	iotsitewise:ListProjects
	iotsitewise:ListTimeSeries
	iotsitewise:PutDefaultEncryptionConfiguration
	iotsitewise:PutLoggingOptions
	iotsitewise:PutStorageConfiguration
	iotsitewise:UpdateAccessPolicy
	iotsitewise:UpdateAsset
	iotsitewise:UpdateAssetModel
	iotsitewise:UpdateAssetModelCompositeModel
	iotsitewise:UpdateAssetProperty
	iotsitewise:UpdateComputationModel
	iotsitewise:UpdateDashboard

服务前缀	操作
	iotsitewise:UpdateDataset iotsitewise:UpdateGateway iotsitewise:UpdateGatewayCapabilityConfiguration iotsitewise:UpdatePortal iotsitewise:UpdateProject

服务前缀	操作
iottwinmaker	iottwinmaker:CancelMetadataTransferJob iottwinmaker:CreateComponentType iottwinmaker:CreateEntity iottwinmaker:CreateMetadataTransferJob iottwinmaker:CreateScene iottwinmaker:CreateSyncJob iottwinmaker:CreateWorkspace iottwinmaker:DeleteComponentType iottwinmaker:DeleteEntity iottwinmaker:DeleteScene iottwinmaker:DeleteSyncJob iottwinmaker:DeleteWorkspace iottwinmaker:ExecuteQuery iottwinmaker:GetMetadataTransferJob iottwinmaker:GetPricingPlan iottwinmaker:GetScene iottwinmaker:GetSyncJob iottwinmaker:ListComponentTypes iottwinmaker:ListComponents iottwinmaker:ListEntities iottwinmaker:ListMetadataTransferJobs

服务前缀	操作
	iottwinmaker:ListProperties iottwinmaker:ListScenes iottwinmaker:ListSyncJobs iottwinmaker:ListSyncResources iottwinmaker:ListWorkspaces iottwinmaker:UpdateComponentType iottwinmaker:UpdateEntity iottwinmaker:UpdatePricingPlan iottwinmaker:UpdateScene iottwinmaker:UpdateWorkspace

服务前缀	操作
iotwireless	iotwireless:AssociateAwsAccountWithPartnerAccount
	iotwireless:AssociateMulticastGroupWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithFuotaTask
	iotwireless:AssociateWirelessDeviceWithMulticastGroup
	iotwireless:AssociateWirelessDeviceWithThing
	iotwireless:AssociateWirelessGatewayWithCertificate
	iotwireless:AssociateWirelessGatewayWithThing
	iotwireless:CancelMulticastGroupSession
	iotwireless:CreateDestination
	iotwireless:CreateDeviceProfile
	iotwireless:CreateFuotaTask
	iotwireless:CreateMulticastGroup
	iotwireless:CreateNetworkAnalyzerConfiguration
	iotwireless:CreateServiceProfile
	iotwireless:CreateWirelessDevice
	iotwireless:CreateWirelessGateway
	iotwireless:CreateWirelessGatewayTask
	iotwireless:CreateWirelessGatewayTaskDefinition
	iotwireless:DeleteDestination
	iotwireless:DeleteDeviceProfile
	iotwireless:DeleteFuotaTask

服务前缀	操作
	iotwireless:DeleteMulticastGroup
	iotwireless:DeleteNetworkAnalyzerConfiguration
	iotwireless:DeleteQueuedMessages
	iotwireless:DeleteServiceProfile
	iotwireless:DeleteWirelessDevice
	iotwireless:DeleteWirelessDeviceImportTask
	iotwireless:DeleteWirelessGateway
	iotwireless:DeleteWirelessGatewayTask
	iotwireless:DeleteWirelessGatewayTaskDefinition
	iotwireless:DeregisterWirelessDevice
	iotwireless:DisassociateAwsAccountFromPartnerAccount
	iotwireless:DisassociateMulticastGroupFromFuotaTask
	iotwireless:DisassociateWirelessDeviceFromFuotaTask
	iotwireless:DisassociateWirelessDeviceFromMulticastGroup
	iotwireless:DisassociateWirelessDeviceFromThing
	iotwireless:DisassociateWirelessGatewayFromCertificate
	iotwireless:DisassociateWirelessGatewayFromThing
	iotwireless:GetDestination
	iotwireless:GetDeviceProfile
	iotwireless:GetEventConfigurationByResourceTypes
	iotwireless:GetFuotaTask

服务前缀	操作
	iotwireless:GetLogLevelsByResourceTypes
	iotwireless:GetMetricConfiguration
	iotwireless:GetMetrics
	iotwireless:GetMulticastGroup
	iotwireless:GetMulticastGroupSession
	iotwireless:GetNetworkAnalyzerConfiguration
	iotwireless:GetPartnerAccount
	iotwireless:GetPosition
	iotwireless:GetPositionConfiguration
	iotwireless:GetPositionEstimate
	iotwireless:GetResourceEventConfiguration
	iotwireless:GetResourceLogLevel
	iotwireless:GetResourcePosition
	iotwireless:GetServiceEndpoint
	iotwireless:GetServiceProfile
	iotwireless:GetWirelessDevice
	iotwireless:GetWirelessDeviceImportTask
	iotwireless:GetWirelessDeviceStatistics
	iotwireless:GetWirelessGateway
	iotwireless:GetWirelessGatewayCertificate
	iotwireless:GetWirelessGatewayFirmwareInformation

服务前缀	操作
	iotwireless:GetWirelessGatewayStatistics
	iotwireless:GetWirelessGatewayTask
	iotwireless:GetWirelessGatewayTaskDefinition
	iotwireless:ListDestinations
	iotwireless:ListDeviceProfiles
	iotwireless:ListDevicesForWirelessDeviceImportTask
	iotwireless:ListEventConfigurations
	iotwireless:ListFuotaTasks
	iotwireless:ListMulticastGroups
	iotwireless:ListMulticastGroupsByFuotaTask
	iotwireless:ListNetworkAnalyzerConfigurations
	iotwireless:ListPartnerAccounts
	iotwireless:ListPositionConfigurations
	iotwireless:ListQueuedMessages
	iotwireless:ListServiceProfiles
	iotwireless:ListWirelessDeviceImportTasks
	iotwireless:ListWirelessDevices
	iotwireless:ListWirelessGatewayTaskDefinitions
	iotwireless:ListWirelessGateways
	iotwireless:PutPositionConfiguration
	iotwireless:PutResourceLogLevel

服务前缀	操作
	iotwireless:ResetAllResourceLogLevels
	iotwireless:ResetResourceLogLevel
	iotwireless:SendDataToMulticastGroup
	iotwireless:SendDataToWirelessDevice
	iotwireless:StartBulkAssociateWirelessDeviceWithMulticastGroup
	iotwireless:StartBulkDisassociateWirelessDeviceFromMulticastGroup
	iotwireless:StartFuotaTask
	iotwireless:StartMulticastGroupSession
	iotwireless:StartNetworkAnalyzerStream
	iotwireless:StartSingleWirelessDeviceImportTask
	iotwireless:StartWirelessDeviceImportTask
	iotwireless:TestWirelessDevice
	iotwireless:UpdateDestination
	iotwireless:UpdateEventConfigurationByResourceTypes
	iotwireless:UpdateFuotaTask
	iotwireless:UpdateLogLevelsByResourceTypes
	iotwireless:UpdateMetricConfiguration
	iotwireless:UpdateMulticastGroup
	iotwireless:UpdateNetworkAnalyzerConfiguration
	iotwireless:UpdatePartnerAccount

服务前缀	操作
	iotwireless:UpdatePosition
	iotwireless:UpdateResourceEventConfiguration
	iotwireless:UpdateResourcePosition
	iotwireless:UpdateWirelessDevice
	iotwireless:UpdateWirelessDeviceImportTask
	iotwireless:UpdateWirelessGateway

服务前缀	操作
ivs	ivs:BatchGetChannel ivs:BatchGetStreamKey ivs:BatchStartViewerSessionRevocation ivs:CreateChannel ivs:CreateEncoderConfiguration ivs:CreateIngestConfiguration ivs:CreateParticipantToken ivs:CreatePlaybackRestrictionPolicy ivs:CreateRecordingConfiguration ivs:CreateStorageConfiguration ivs:CreateStreamKey ivs:DeleteChannel ivs:DeleteEncoderConfiguration ivs:DeleteIngestConfiguration ivs:DeletePlaybackKeyPair ivs:DeletePlaybackRestrictionPolicy ivs:DeletePublicKey ivs:DeleteRecordingConfiguration ivs:DeleteStorageConfiguration ivs:DeleteStreamKey ivs:DisconnectParticipant

服务前缀	操作
	ivs:GetChannel ivs:GetComposition ivs:GetEncoderConfiguration ivs:GetIngestConfiguration ivs:GetParticipant ivs:GetPlaybackKeyPair ivs:GetPlaybackRestrictionPolicy ivs:GetPublicKey ivs:GetRecordingConfiguration ivs:GetStorageConfiguration ivs:GetStream ivs:GetStreamKey ivs:GetStreamSession ivs:ImportPlaybackKeyPair ivs:ImportPublicKey ivs:ListChannels ivs:ListCompositions ivs:ListEncoderConfigurations ivs:ListIngestConfigurations ivs:ListParticipantEvents ivs:ListParticipantReplicas

服务前缀	操作
	ivs:ListParticipants ivs:ListPlaybackKeyPairs ivs:ListPlaybackRestrictionPolicies ivs:ListPublicKeys ivs:ListRecordingConfigurations ivs:ListStorageConfigurations ivs:ListStreamKeys ivs:ListStreamSessions ivs:ListStreams ivs:PutMetadata ivs:StartComposition ivs:StartViewerSessionRevocation ivs:StopComposition ivs:StopStream ivs:UpdateChannel ivs:UpdateIngestConfiguration ivs:UpdatePlaybackRestrictionPolicy

服务前缀	操作
ivschat	ivschat:CreateChatToken ivschat:CreateLoggingConfiguration ivschat:CreateRoom ivschat:DeleteLoggingConfiguration ivschat:DeleteMessage ivschat:DeleteRoom ivschat:DisconnectUser ivschat:GetLoggingConfiguration ivschat:GetRoom ivschat:ListLoggingConfigurations ivschat:ListRooms ivschat:SendEvent ivschat:UpdateLoggingConfiguration ivschat:UpdateRoom

服务前缀	操作
kafka	kafka:BatchAssociateScramSecret kafka:BatchDisassociateScramSecret kafka:CreateCluster kafka:CreateClusterV2 kafka:CreateConfiguration kafka:CreateReplicator kafka:CreateVpcConnection kafka:DeleteCluster kafka:DeleteClusterPolicy kafka:DeleteConfiguration kafka:DeleteReplicator kafka:DeleteVpcConnection kafka:DescribeCluster kafka:DescribeClusterOperation kafka:DescribeClusterOperationV2 kafka:DescribeClusterV2 kafka:DescribeConfiguration kafka:DescribeConfigurationRevision kafka:DescribeVpcConnection kafka:GetBootstrapBrokers kafka:GetClusterPolicy

服务前缀	操作
	kafka:GetCompatibleKafkaVersions kafka:ListClientVpcConnections kafka:ListClusterOperations kafka:ListClusterOperationsV2 kafka:ListClusters kafka:ListClustersV2 kafka:ListConfigurationRevisions kafka:ListConfigurations kafka:ListKafkaVersions kafka:ListNodes kafka:ListReplicators kafka:ListScramSecrets kafka:ListVpcConnections kafka:PutClusterPolicy kafka:RebootBroker kafka:RejectClientVpcConnection kafka:UpdateBrokerCount kafka:UpdateBrokerStorage kafka:UpdateBrokerType kafka:UpdateClusterConfiguration kafka:UpdateClusterKafkaVersion

服务前缀	操作
	kafka:UpdateConfiguration kafka:UpdateConnectivity kafka:UpdateMonitoring kafka:UpdateRebalancing kafka:UpdateReplicationInfo kafka:UpdateSecurity kafka:UpdateStorage
kafkaconnect	kafkaconnect:CreateConnector kafkaconnect:CreateCustomPlugin kafkaconnect:CreateWorkerConfiguration kafkaconnect:DeleteConnector kafkaconnect:DeleteCustomPlugin kafkaconnect:DeleteWorkerConfiguration kafkaconnect:DescribeConnector kafkaconnect:DescribeCustomPlugin kafkaconnect:DescribeWorkerConfiguration kafkaconnect:ListConnectorOperations kafkaconnect:ListConnectors kafkaconnect:ListCustomPlugins kafkaconnect:ListWorkerConfigurations kafkaconnect:UpdateConnector

服务前缀	操作
kendra	kendra:AssociateEntitiesToExperience kendra:AssociatePersonasToEntities kendra:BatchDeleteDocument kendra:BatchDeleteFeaturedResultsSet kendra:BatchGetDocumentStatus kendra:BatchPutDocument kendra:ClearQuerySuggestions kendra:CreateAccessControlConfiguration kendra:CreateDataSource kendra:CreateExperience kendra:CreateFaq kendra:CreateFeaturedResultsSet kendra:CreateIndex kendra:CreateQuerySuggestionsBlockList kendra:CreateThesaurus kendra:DeleteDataSource kendra:DeleteExperience kendra:DeleteFaq kendra:DeleteIndex kendra:DeletePrincipalMapping kendra:DeleteQuerySuggestionsBlockList

服务前缀	操作
	kendra:DeleteThesaurus kendra:DescribeAccessControlConfiguration kendra:DescribeDataSource kendra:DescribeExperience kendra:DescribeFaq kendra:DescribeFeaturedResultsSet kendra:DescribeIndex kendra:DescribePrincipalMapping kendra:DescribeQuerySuggestionsBlockList kendra:DescribeQuerySuggestionsConfig kendra:DescribeThesaurus kendra:DisassociateEntitiesFromExperience kendra:DisassociatePersonasFromEntities kendra:GetQuerySuggestions kendra:GetSnapshots kendra:ListAccessControlConfigurations kendra:ListDataSourceSyncJobs kendra:ListDataSources kendra:ListEntityPersonas kendra:ListExperienceEntities kendra:ListExperiences

服务前缀	操作
	kendra:ListFaqs kendra:ListFeaturedResultsSets kendra:ListGroupsOlderThanOrderingId kendra:ListIndices kendra:ListQuerySuggestionsBlockLists kendra:ListThesauri kendra:PutPrincipalMapping kendra:Query kendra:Retrieve kendra:StartDataSourceSyncJob kendra:StopDataSourceSyncJob kendra:SubmitFeedback kendra:UpdateDataSource kendra:UpdateExperience kendra:UpdateFeaturedResultsSet kendra:UpdateIndex kendra:UpdateQuerySuggestionsBlockList kendra:UpdateQuerySuggestionsConfig kendra:UpdateThesaurus

服务前缀	操作
kinesis	kinesis:CreateStream
	kinesis:DecreaseStreamRetentionPeriod
	kinesis:DeleteStream
	kinesis:DeregisterStreamConsumer
	kinesis:DescribeAccountSettings
	kinesis:DescribeLimits
	kinesis:DescribeStream
	kinesis:DescribeStreamConsumer
	kinesis:DescribeStreamSummary
	kinesis:DisableEnhancedMonitoring
	kinesis:EnableEnhancedMonitoring
	kinesis:IncreaseStreamRetentionPeriod
	kinesis:ListShards
	kinesis:ListStreamConsumers
	kinesis:ListStreams
	kinesis:MergeShards
	kinesis:RegisterStreamConsumer
	kinesis:SplitShard
	kinesis:StartStreamEncryption
	kinesis:StopStreamEncryption
	kinesis:UpdateAccountSettings

服务前缀	操作
	kinesis:UpdateShardCount kinesis:UpdateStreamMode

服务前缀	操作
kinesisanalytics	kinesisanalytics:AddApplicationCloudWatchLoggingOption kinesisanalytics:AddApplicationInput kinesisanalytics:AddApplicationInputProcessingConfiguration kinesisanalytics:AddApplicationOutput kinesisanalytics:AddApplicationReferenceDataSource kinesisanalytics:AddApplicationVpcConfiguration kinesisanalytics:CreateApplication kinesisanalytics:CreateApplicationPresignedUrl kinesisanalytics:CreateApplicationSnapshot kinesisanalytics:DeleteApplication kinesisanalytics:DeleteApplicationCloudWatchLoggingOption kinesisanalytics:DeleteApplicationInputProcessingConfiguration kinesisanalytics:DeleteApplicationOutput kinesisanalytics:DeleteApplicationReferenceDataSource kinesisanalytics:DeleteApplicationSnapshot kinesisanalytics:DeleteApplicationVpcConfiguration kinesisanalytics:DescribeApplication kinesisanalytics:DescribeApplicationOperation kinesisanalytics:DescribeApplicationSnapshot kinesisanalytics:DescribeApplicationVersion kinesisanalytics:DiscoverInputSchema

服务前缀	操作
	kinesisanalytics:ListApplicationOperations kinesisanalytics:ListApplicationSnapshots kinesisanalytics:ListApplicationVersions kinesisanalytics:ListApplications kinesisanalytics:RollbackApplication kinesisanalytics:StartApplication kinesisanalytics:StopApplication kinesisanalytics:UpdateApplication kinesisanalytics:UpdateApplicationMaintenanceConfiguration

服务前缀	操作
kms	kms:CancelKeyDeletion
	kms:ConnectCustomKeyStore
	kms:CreateAlias
	kms:CreateCustomKeyStore
	kms:CreateGrant
	kms:CreateKey
	kms:Decrypt
	kms:DeleteAlias
	kms:DeleteCustomKeyStore
	kms:DeleteImportedKeyMaterial
	kms:DeriveSharedSecret
	kms:DescribeCustomKeyStores
	kms:DescribeKey
	kms:DisableKey
	kms:DisableKeyRotation
	kms:DisconnectCustomKeyStore
	kms:EnableKey
	kms:EnableKeyRotation
	kms:Encrypt
	kms:GenerateDataKey
	kms:GenerateDataKeyPair

服务前缀	操作
	kms:GenerateDataKeyPairWithoutPlaintext
	kms:GenerateDataKeyWithoutPlaintext
	kms:GenerateMac
	kms:GenerateRandom
	kms:GetKeyPolicy
	kms:GetKeyRotationStatus
	kms:GetParametersForImport
	kms:GetPublicKey
	kms:ImportKeyMaterial
	kms:ListAliases
	kms:ListGrants
	kms:ListKeyPolicies
	kms:ListKeyRotations
	kms:ListKeys
	kms:ListRetirableGrants
	kms:ReplicateKey
	kms:RetireGrant
	kms:RevokeGrant
	kms:RotateKeyOnDemand
	kms:ScheduleKeyDeletion
	kms:Sign

服务前缀	操作
	kms:UpdateAlias kms:UpdateCustomKeyStore kms:UpdateKeyDescription kms:UpdatePrimaryRegion kms:Verify kms:VerifyMac

服务前缀	操作
lambda	lambda:AddLayerVersionPermission lambda:AddPermission lambda:CreateAlias lambda:CreateCodeSigningConfig lambda:CreateEventSourceMapping lambda:CreateFunction lambda:CreateFunctionUrlConfig lambda:DeleteAlias lambda:DeleteCapacityProvider lambda:DeleteCodeSigningConfig lambda:DeleteEventSourceMapping lambda:DeleteFunction lambda:DeleteFunctionCodeSigningConfig lambda:DeleteFunctionConcurrency lambda:DeleteFunctionEventInvokeConfig lambda:DeleteFunctionUrlConfig lambda:DeleteLayerVersion lambda:DeleteProvisionedConcurrencyConfig lambda:GetAccountSettings lambda:GetAlias lambda:GetCapacityProvider

服务前缀	操作
	lambda:GetCodeSigningConfig lambda:GetEventSourceMapping lambda:GetFunction lambda:GetFunctionCodeSigningConfig lambda:GetFunctionConcurrency lambda:GetFunctionConfiguration lambda:GetFunctionEventInvokeConfig lambda:GetFunctionRecursionConfig lambda:GetFunctionScalingConfig lambda:GetFunctionUrlConfig lambda:GetLayerVersion lambda:GetLayerVersionPolicy lambda:GetPolicy lambda:GetProvisionedConcurrencyConfig lambda:GetRuntimeManagementConfig lambda:ListAliases lambda:ListCapacityProviders lambda:ListCodeSigningConfigs lambda:ListDurableExecutionsByFunction lambda:ListEventSourceMappings lambda:ListFunctionEventInvokeConfigs

服务前缀	操作
	lambda:ListFunctionUrlConfigs lambda:ListFunctions lambda:ListFunctionsByCodeSigningConfig lambda:ListLayerVersions lambda:ListLayers lambda:ListProvisionedConcurrencyConfigs lambda:ListVersionsByFunction lambda:PublishLayerVersion lambda:PublishVersion lambda:PutFunctionCodeSigningConfig lambda:PutFunctionConcurrency lambda:PutFunctionEventInvokeConfig lambda:PutFunctionRecursionConfig lambda:PutFunctionScalingConfig lambda:PutProvisionedConcurrencyConfig lambda:PutRuntimeManagementConfig lambda:RemoveLayerVersionPermission lambda:RemovePermission lambda:UpdateAlias lambda:UpdateCapacityProvider lambda:UpdateCodeSigningConfig

服务前缀	操作
	lambda:UpdateEventSourceMapping lambda:UpdateFunctionCode lambda:UpdateFunctionConfiguration lambda:UpdateFunctionEventInvokeConfig lambda:UpdateFunctionUrlConfig

服务前缀	操作
lex	lex:BatchCreateCustomVocabularyItem
	lex:BatchDeleteCustomVocabularyItem
	lex:BatchUpdateCustomVocabularyItem
	lex:BuildBotLocale
	lex:CreateBotAlias
	lex:CreateBotReplica
	lex:CreateBotVersion
	lex:CreateExport
	lex:CreateIntentVersion
	lex:CreateResourcePolicy
	lex:CreateSlotTypeVersion
	lex:CreateTestSetDiscrepancyReport
	lex:CreateUploadUrl
	lex:DeleteBot
	lex:DeleteBotChannelAssociation
	lex:DeleteBotReplica
	lex:DeleteExport
	lex:DeleteImport
	lex:DeleteIntentVersion
	lex:DeleteResourcePolicy
	lex:DeleteSlotTypeVersion

服务前缀	操作
	lex:DeleteTestSet
	lex:DeleteUtterances
	lex:DescribeBotAlias
	lex:DescribeBotRecommendation
	lex:DescribeBotReplica
	lex:DescribeBotResourceGeneration
	lex:DescribeBotVersion
	lex:DescribeCustomVocabularyMetadata
	lex:DescribeExport
	lex:DescribeImport
	lex:DescribeResourcePolicy
	lex:DescribeTestExecution
	lex:DescribeTestSet
	lex:DescribeTestSetDiscrepancyReport
	lex:DescribeTestSetGeneration
	lex:GenerateBotElement
	lex:GetBot
	lex:GetBotAlias
	lex:GetBotAliases
	lex:GetBotChannelAssociation
	lex:GetBotChannelAssociations

服务前缀	操作
	lex:GetBotVersions
	lex:GetBots
	lex:GetBuiltinIntent
	lex:GetBuiltinIntents
	lex:GetBuiltinSlotTypes
	lex:GetExport
	lex:GetImport
	lex:GetIntent
	lex:GetIntentVersions
	lex:GetIntents
	lex:GetMigration
	lex:GetMigrations
	lex:GetSlotType
	lex:GetSlotTypeVersions
	lex:GetSlotTypes
	lex:GetTestExecutionArtifactsUrl
	lex:GetUtterancesView
	lex:ListBotAliasReplicas
	lex:ListBotAliases
	lex:ListBotRecommendations
	lex:ListBotReplicas

服务前缀	操作
	lex:ListBotResourceGenerations
	lex:ListBotVersionReplicas
	lex:ListBotVersions
	lex:ListBots
	lex:ListBuiltInIntents
	lex:ListBuiltInSlotTypes
	lex:ListCustomVocabularyItems
	lex:ListExports
	lex:ListImports
	lex:ListIntentMetrics
	lex:ListIntentPaths
	lex:ListRecommendedIntents
	lex:ListSessionAnalyticsData
	lex:ListSessionMetrics
	lex:ListTestExecutionResultItems
	lex:ListTestExecutions
	lex:ListTestSets
	lex:PutBot
	lex:PutBotAlias
	lex:PutIntent
	lex:PutSlotType

服务前缀	操作
	lex:SearchAssociatedTranscripts lex:StartBotRecommendation lex:StartImport lex:StartMigration lex:StartTestExecution lex:StartTestSetGeneration lex:StopBotRecommendation lex:UpdateBotAlias lex:UpdateBotRecommendation lex:UpdateExport lex:UpdateResourcePolicy
license-manager-linux-subscriptions	license-manager-linux-subscriptions:DeregisterSubscriptionProvider license-manager-linux-subscriptions:GetRegisteredSubscriptionProvider license-manager-linux-subscriptions:GetServiceSettings license-manager-linux-subscriptions:ListLinuxSubscriptionInstances license-manager-linux-subscriptions:ListLinuxSubscriptions license-manager-linux-subscriptions:ListRegisteredSubscriptionProviders license-manager-linux-subscriptions:RegisterSubscriptionProvider license-manager-linux-subscriptions:UpdateServiceSettings

服务前缀	操作
lightsail	lightsail:AllocateStaticIp
	lightsail:AttachCertificateToDistribution
	lightsail:AttachDisk
	lightsail:AttachInstancesToLoadBalancer
	lightsail:AttachLoadBalancerTlsCertificate
	lightsail:AttachStaticIp
	lightsail:CloseInstancePublicPorts
	lightsail:CopySnapshot
	lightsail:CreateBucket
	lightsail:CreateBucketAccessKey
	lightsail:CreateCertificate
	lightsail:CreateCloudFormationStack
	lightsail:CreateContactMethod
	lightsail:CreateContainerService
	lightsail:CreateContainerServiceDeployment
	lightsail:CreateContainerServiceRegistryLogin
	lightsail:CreateDisk
	lightsail:CreateDiskFromSnapshot
	lightsail:CreateDiskSnapshot
	lightsail:CreateDistribution
	lightsail:CreateDomain

服务前缀	操作
	lightsail:CreateGUISessionAccessDetails
	lightsail:CreateInstanceSnapshot
	lightsail:CreateInstances
	lightsail:CreateInstancesFromSnapshot
	lightsail:CreateKeyPair
	lightsail:CreateLoadBalancer
	lightsail:CreateLoadBalancerTlsCertificate
	lightsail:CreateRelationalDatabase
	lightsail:CreateRelationalDatabaseFromSnapshot
	lightsail:CreateRelationalDatabaseSnapshot
	lightsail:DeleteAlarm
	lightsail:DeleteAutoSnapshot
	lightsail:DeleteBucket
	lightsail:DeleteBucketAccessKey
	lightsail:DeleteCertificate
	lightsail:DeleteContactMethod
	lightsail:DeleteContainerImage
	lightsail:DeleteContainerService
	lightsail:DeleteDisk
	lightsail:DeleteDiskSnapshot
	lightsail:DeleteDistribution

服务前缀	操作
	lightsail:DeleteDomain
	lightsail:DeleteDomainEntry
	lightsail:DeleteInstance
	lightsail:DeleteInstanceSnapshot
	lightsail:DeleteKeyPair
	lightsail:DeleteKnownHostKeys
	lightsail:DeleteLoadBalancer
	lightsail:DeleteLoadBalancerTlsCertificate
	lightsail:DeleteRelationalDatabase
	lightsail:DeleteRelationalDatabaseSnapshot
	lightsail:DetachCertificateFromDistribution
	lightsail:DetachDisk
	lightsail:DetachInstancesFromLoadBalancer
	lightsail:DetachStaticIp
	lightsail:DisableAddOn
	lightsail:DownloadDefaultKeyPair
	lightsail:EnableAddOn
	lightsail:ExportSnapshot
	lightsail:GetActiveNames
	lightsail:GetAlarms
	lightsail:GetAutoSnapshots

服务前缀	操作
	lightsail:GetBlueprints
	lightsail:GetBucketAccessKeys
	lightsail:GetBucketBundles
	lightsail:GetBucketMetricData
	lightsail:GetBuckets
	lightsail:GetBundles
	lightsail:GetCertificates
	lightsail:GetCloudFormationStackRecords
	lightsail:GetContactMethods
	lightsail:GetContainerAPIMetadata
	lightsail:GetContainerImages
	lightsail:GetContainerLog
	lightsail:GetContainerServiceDeployments
	lightsail:GetContainerServiceMetricData
	lightsail:GetContainerServicePowers
	lightsail:GetContainerServices
	lightsail:GetCostEstimate
	lightsail:GetDisk
	lightsail:GetDiskSnapshot
	lightsail:GetDiskSnapshots
	lightsail:GetDisks

服务前缀	操作
	lightsail:GetDistributionBundles
	lightsail:GetDistributionLatestCacheReset
	lightsail:GetDistributionMetricData
	lightsail:GetDistributions
	lightsail:GetDomain
	lightsail:GetExportSnapshotRecords
	lightsail:GetInstance
	lightsail:GetInstanceMetricData
	lightsail:GetInstancePortStates
	lightsail:GetInstanceSnapshot
	lightsail:GetInstanceSnapshots
	lightsail:GetInstanceState
	lightsail:GetInstances
	lightsail:GetKeyPair
	lightsail:GetKeyPairs
	lightsail:GetLoadBalancer
	lightsail:GetLoadBalancerMetricData
	lightsail:GetLoadBalancerTlsCertificates
	lightsail:GetLoadBalancerTlsPolicies
	lightsail:GetLoadBalancers
	lightsail:GetOperation

服务前缀	操作
	lightsail:GetOperations
	lightsail:GetOperationsForResource
	lightsail:GetRegions
	lightsail:GetRelationalDatabase
	lightsail:GetRelationalDatabaseBlueprints
	lightsail:GetRelationalDatabaseBundles
	lightsail:GetRelationalDatabaseEvents
	lightsail:GetRelationalDatabaseLogEvents
	lightsail:GetRelationalDatabaseLogStreams
	lightsail:GetRelationalDatabaseMasterUserPassword
	lightsail:GetRelationalDatabaseMetricData
	lightsail:GetRelationalDatabaseParameters
	lightsail:GetRelationalDatabaseSnapshot
	lightsail:GetRelationalDatabaseSnapshots
	lightsail:GetRelationalDatabases
	lightsail:GetSetupHistory
	lightsail:GetStaticIp
	lightsail:GetStaticIps
	lightsail:ImportKeyPair
	lightsail:IsVpcPeered
	lightsail:OpenInstancePublicPorts

服务前缀	操作
	lightsail:PeerVpc
	lightsail:PutAlarm
	lightsail:PutInstancePublicPorts
	lightsail:RebootInstance
	lightsail:RebootRelationalDatabase
	lightsail:RegisterContainerImage
	lightsail:releaseStatiCip
	lightsail:ResetDistributionCache
	lightsail:SendContactMethodVerification
	lightsail:SetIpAddressType
	lightsail:SetResourceAccessForBucket
	lightsail:SetupInstanceHttps
	lightsail:StartGUISession
	lightsail:StartInstance
	lightsail:StartRelationalDatabase
	lightsail:StopGUISession
	lightsail:StopInstance
	lightsail:StopRelationalDatabase
	lightsail:TestAlarm
	lightsail:UnpeerVpc
	lightsail:UpdateBucket

服务前缀	操作
	lightsail:UpdateBucketBundle lightsail:UpdateContainerService lightsail:UpdateDistribution lightsail:UpdateDistributionBundle lightsail:UpdateDomainEntry lightsail:UpdateInstanceMetadataOptions lightsail:UpdateLoadBalancerAttribute lightsail:UpdateRelationalDatabase lightsail:UpdateRelationalDatabaseParameters

服务前缀	操作
logs	logs:AssociateKmsKey logs:AssociateSourceToS3TableIntegration logs:CancelExportTask logs:CancelImportTask logs:CreateDelivery logs:CreateExportTask logs:CreateLogAnomalyDetector logs:CreateLogGroup logs:CreateLogStream logs:DeleteDataProtectionPolicy logs:DeleteDelivery logs:DeleteDeliveryDestination logs:DeleteDeliveryDestinationPolicy logs:DeleteDeliverySource logs:DeleteDestination logs:DeleteIndexPolicy logs:DeleteIntegration logs:DeleteLogAnomalyDetector logs:DeleteLogGroup logs:DeleteLogStream logs:DeleteMetricFilter

服务前缀	操作
	logs:DeleteQueryDefinition logs:DeleteResourcePolicy logs:DeleteRetentionPolicy logs:DeleteScheduledQuery logs:DeleteSubscriptionFilter logs:DeleteTransformer logs:DescribeAccountPolicies logs:DescribeConfigurationTemplates logs:DescribeDeliveries logs:DescribeDeliveryDestinations logs:DescribeDeliverySources logs:DescribeDestinations logs:DescribeExportTasks logs:DescribeFieldIndexes logs:DescribeImportTaskBatches logs:DescribeImportTasks logs:DescribeIndexPolicies logs:DescribeLogGroups logs:DescribeLogStreams logs:DescribeMetricFilters logs:DescribeQueries

服务前缀	操作
	logs:DescribeQueryDefinitions logs:DescribeResourcePolicies logs:DescribeSubscriptionFilters logs:DisassociateKmsKey logs:DisassociateSourceFromS3TableIntegration logs:GetDataProtectionPolicy logs:GetDelivery logs:GetDeliveryDestination logs:GetDeliveryDestinationPolicy logs:GetDeliverySource logs:GetIntegration logs:GetLogAnomalyDetector logs:GetLogFields logs:GetLogGroupFields logs:GetLogRecord logs:GetQueryResults logs:GetScheduledQuery logs:GetScheduledQueryHistory logs:GetTransformer logs:ListAnomalies logs:ListIntegrations

服务前缀	操作
	logs:ListLogAnomalyDetectors logs:ListLogGroupsForQuery logs:ListScheduledQueries logs:ListSourcesForS3TableIntegration logs:PutDataProtectionPolicy logs:PutDeliveryDestination logs:PutDeliveryDestinationPolicy logs:PutDeliverySource logs:PutDestination logs:PutDestinationPolicy logs:PutIndexPolicy logs:PutIntegration logs:PutLogGroupDeletionProtection logs:PutMetricFilter logs:PutQueryDefinition logs:PutResourcePolicy logs:PutRetentionPolicy logs:PutSubscriptionFilter logs:PutTransformer logs:StartLiveTail logs:StartQuery

服务前缀	操作
	logs:StopQuery logs:TestMetricFilter logs:TestTransformer logs:UpdateAnomaly logs:UpdateDeliveryConfiguration logs:UpdateLogAnomalyDetector

服务前缀	操作
lookoutequipment	lookoutequipment:CreateDataset lookoutequipment:CreateInferenceScheduler lookoutequipment:CreateLabel lookoutequipment:CreateLabelGroup lookoutequipment:CreateModel lookoutequipment:DeleteDataset lookoutequipment:DeleteInferenceScheduler lookoutequipment:DeleteLabel lookoutequipment:DeleteLabelGroup lookoutequipment:DeleteModel lookoutequipment:DeleteResourcePolicy lookoutequipment:DeleteRetrainingScheduler lookoutequipment:DescribeDataIngestionJob lookoutequipment:DescribeDataset lookoutequipment:DescribeInferenceScheduler lookoutequipment:DescribeLabelGroup lookoutequipment:DescribeModel lookoutequipment:DescribeModelVersion lookoutequipment:DescribeResourcePolicy lookoutequipment:DescribeRetrainingScheduler lookoutequipment:Describelabel

服务前缀	操作
	lookoutequipment:ImportDataset
	lookoutequipment:ImportModelVersion
	lookoutequipment:ListDataIngestionJobs
	lookoutequipment:ListDatasets
	lookoutequipment:ListInferenceEvents
	lookoutequipment:ListInferenceExecutions
	lookoutequipment:ListInferenceSchedulers
	lookoutequipment:ListLabelGroups
	lookoutequipment:ListLabels
	lookoutequipment:ListModelVersions
	lookoutequipment:ListModels
	lookoutequipment:ListRetrainingSchedulers
	lookoutequipment:ListSensorStatistics
	lookoutequipment:PutResourcePolicy
	lookoutequipment:StartDataIngestionJob
	lookoutequipment:StartInferenceScheduler
	lookoutequipment:StartRetrainingScheduler
	lookoutequipment:StopInferenceScheduler
	lookoutequipment:StopRetrainingScheduler
	lookoutequipment:UpdateActiveModelVersion
	lookoutequipment:UpdateInferenceScheduler

服务前缀	操作
	lookoutequipment:UpdateLabelGroup lookoutequipment:UpdateModel lookoutequipment:UpdateRetrainingScheduler

服务前缀	操作
lookoutmetrics	lookoutmetrics:ActivateAnomalyDetector lookoutmetrics:BackTestAnomalyDetector lookoutmetrics:CreateAlert lookoutmetrics:CreateAnomalyDetector lookoutmetrics:CreateMetricSet lookoutmetrics:DeactivateAnomalyDetector lookoutmetrics:DeleteAlert lookoutmetrics:DeleteAnomalyDetector lookoutmetrics:DescribeAlert lookoutmetrics:DescribeAnomalyDetectionExecutions lookoutmetrics:DescribeAnomalyDetector lookoutmetrics:DescribeMetricSet lookoutmetrics:DetectMetricSetConfig lookoutmetrics:GetAnomalyGroup lookoutmetrics:GetDataQualityMetrics lookoutmetrics:GetFeedback lookoutmetrics:GetSampleData lookoutmetrics:ListAlerts lookoutmetrics:ListAnomalyDetectors lookoutmetrics:ListAnomalyGroupRelatedMetrics lookoutmetrics:ListAnomalyGroupSummaries

服务前缀	操作
	lookoutmetrics:ListAnomalyGroupTimeSeries lookoutmetrics:ListMetricSets lookoutmetrics:PutFeedback lookoutmetrics:UpdateAlert lookoutmetrics:UpdateAnomalyDetector lookoutmetrics:UpdateMetricSet

服务前缀	操作
lookoutvision	lookoutvision:CreateDataset lookoutvision:CreateModel lookoutvision:CreateProject lookoutvision:DeleteDataset lookoutvision:DeleteModel lookoutvision:DeleteProject lookoutvision:DescribeDataset lookoutvision:DescribeModel lookoutvision:DescribeModelPackagingJob lookoutvision:DescribeProject lookoutvision:DetectAnomalies lookoutvision:ListDatasetEntries lookoutvision:ListModelPackagingJobs lookoutvision:ListModels lookoutvision:ListProjects lookoutvision:StartModel lookoutvision:StartModelPackagingJob lookoutvision:StopModel lookoutvision:UpdateDatasetEntries

服务前缀	操作
m2	m2:CancelBatchJobExecution m2:CreateApplication m2:CreateDataSetExportTask m2:CreateDataSetImportTask m2:CreateDeployment m2:CreateEnvironment m2:DeleteApplication m2:DeleteApplicationFromEnvironment m2:DeleteEnvironment m2:GetApplication m2:GetApplicationVersion m2:GetBatchJobExecution m2:GetDataSetDetails m2:GetDataSetExportTask m2:GetDataSetImportTask m2:GetDeployment m2:GetEnvironment m2:GetSignedBluinsightsUrl m2:ListApplicationVersions m2:ListApplications m2:ListBatchJobDefinitions

服务前缀	操作
	m2:ListBatchJobExecutions m2:ListBatchJobRestartPoints m2:ListDataSetExportHistory m2:ListDataSetImportHistory m2:ListDataSets m2:ListDeployments m2:ListEngineVersions m2:ListEnvironments m2:StartApplication m2:StartBatchJob m2:StopApplication m2:UpdateApplication m2:UpdateEnvironment

服务前缀	操作
managedblockchain	managedblockchain:CreateAccessor
	managedblockchain:CreateMember
	managedblockchain:CreateNetwork
	managedblockchain:CreateNode
	managedblockchain:CreateProposal
	managedblockchain:DeleteAccessor
	managedblockchain:DeleteMember
	managedblockchain:DeleteNode
	managedblockchain:GetAccessor
	managedblockchain:GetMember
	managedblockchain:GetNetwork
	managedblockchain:GetNode
	managedblockchain:GetProposal
	managedblockchain:InvokeRpcPolygonMainnet
	managedblockchain:InvokeRpcPolygonMumbaiTestnet
	managedblockchain:ListAccessors
	managedblockchain:ListInvitations
	managedblockchain:ListMembers
	managedblockchain:ListNetworks
	managedblockchain:ListNodes
	managedblockchain:ListProposalVotes

服务前缀	操作
	managedblockchain:ListProposals
	managedblockchain:RejectInvitation
	managedblockchain:UpdateMember
	managedblockchain:UpdateNode
	managedblockchain:VoteOnProposal

服务前缀	操作
mediaconnect	mediaconnect:AddBridgeOutputs mediaconnect:AddBridgeSources mediaconnect:AddFlowMediaStreams mediaconnect:AddFlowOutputs mediaconnect:AddFlowSources mediaconnect:AddFlowVpcInterfaces mediaconnect:CreateBridge mediaconnect:CreateFlow mediaconnect:CreateGateway mediaconnect:DeleteBridge mediaconnect:DeleteFlow mediaconnect:DeleteGateway mediaconnect:DeleteRouterInput mediaconnect:DeleteRouterNetworkInterface mediaconnect:DeleteRouterOutput mediaconnect:DeregisterGatewayInstance mediaconnect:DescribeBridge mediaconnect:DescribeFlow mediaconnect:DescribeFlowSourceMetadata mediaconnect:DescribeFlowSourceThumbnail mediaconnect:DescribeGateway

服务前缀	操作
	mediaconnect:DescribeGatewayInstance mediaconnect:DescribeOffering mediaconnect:DescribeReservation mediaconnect:GetRouterInput mediaconnect:GetRouterInputSourceMetadata mediaconnect:GetRouterInputThumbnail mediaconnect:GetRouterNetworkInterface mediaconnect:GetRouterOutput mediaconnect:GrantFlowEntitlements mediaconnect:ListBridges mediaconnect:ListEntitlements mediaconnect:ListFlows mediaconnect:ListGatewayInstances mediaconnect:ListGateways mediaconnect:ListOfferings mediaconnect:ListReservations mediaconnect:ListRouterInputs mediaconnect:ListRouterNetworkInterfaces mediaconnect:ListRouterOutputs mediaconnect:PurchaseOffering mediaconnect:RemoveBridgeOutput

服务前缀	操作
	mediaconnect:RemoveBridgeSource mediaconnect:RemoveFlowMediaStream mediaconnect:RemoveFlowOutput mediaconnect:RemoveFlowSource mediaconnect:RemoveFlowVpcInterface mediaconnect:RestartRouterInput mediaconnect:RestartRouterOutput mediaconnect:RevokeFlowEntitlement mediaconnect:StartFlow mediaconnect:StartRouterInput mediaconnect:StartRouterOutput mediaconnect:StopFlow mediaconnect:StopRouterInput mediaconnect:StopRouterOutput mediaconnect:TakeRouterInput mediaconnect:UpdateBridge mediaconnect:UpdateBridgeOutput mediaconnect:UpdateBridgeSource mediaconnect:UpdateBridgeState mediaconnect:UpdateFlow mediaconnect:UpdateFlowEntitlement

服务前缀	操作
	mediaconnect:UpdateFlowMediaStream mediaconnect:UpdateGatewayInstance

服务前缀	操作
mediaconvert	mediaconvert:AssociateCertificate
	mediaconvert:CancelJob
	mediaconvert:CreateJob
	mediaconvert:CreateJobTemplate
	mediaconvert:CreatePreset
	mediaconvert:CreateQueue
	mediaconvert:CreateResourceShare
	mediaconvert:DeleteJobTemplate
	mediaconvert:DeletePolicy
	mediaconvert:DeletePreset
	mediaconvert:DeleteQueue
	mediaconvert:DescribeEndpoints
	mediaconvert:DisassociateCertificate
	mediaconvert:GetJob
	mediaconvert:GetJobTemplate
	mediaconvert:GetPolicy
	mediaconvert:GetPreset
	mediaconvert:GetQueue
	mediaconvert:ListJobTemplates
	mediaconvert:ListJobs
	mediaconvert:ListPresets

服务前缀	操作
	mediaconvert:ListQueues
	mediaconvert:ListVersions
	mediaconvert:Probe
	mediaconvert:PutPolicy
	mediaconvert:SearchJobs
	mediaconvert:UpdateJobTemplate
	mediaconvert:UpdatePreset
	mediaconvert:UpdateQueue

服务前缀	操作
medialive	medialive:AcceptInputDeviceTransfer medialive:BatchDelete medialive:BatchStart medialive:BatchStop medialive:BatchUpdateSchedule medialive:CancelInputDeviceTransfer medialive:ClaimDevice medialive:CreateChannel medialive:CreateChannelPlacementGroup medialive:CreateCloudWatchAlarmTemplate medialive:CreateCloudWatchAlarmTemplateGroup medialive:CreateCluster medialive:CreateEventBridgeRuleTemplate medialive:CreateEventBridgeRuleTemplateGroup medialive:CreateInput medialive:CreateInputSecurityGroup medialive:CreateMultiplex medialive:CreateMultiplexProgram medialive:CreateNetwork medialive:CreateNode medialive:CreateNodeRegistrationScript

服务前缀	操作
	medialive:CreatePartnerInput medialive:CreateSdiSource medialive:CreateSignalMap medialive:DeleteChannel medialive:DeleteChannelPlacementGroup medialive:DeleteCloudWatchAlarmTemplate medialive:DeleteCloudWatchAlarmTemplateGroup medialive:DeleteCluster medialive:DeleteEventBridgeRuleTemplate medialive:DeleteEventBridgeRuleTemplateGroup medialive:DeleteInput medialive:DeleteInputSecurityGroup medialive:DeleteMultiplex medialive:DeleteMultiplexProgram medialive:DeleteNetwork medialive:DeleteNode medialive:DeleteReservation medialive:DeleteSchedule medialive:DeleteSdiSource medialive:DeleteSignalMap medialive:DescribeAccountConfiguration

服务前缀	操作
	medialive:DescribeChannel medialive:DescribeChannelPlacementGroup medialive:DescribeCluster medialive:DescribeInput medialive:DescribeInputDevice medialive:DescribeInputDeviceThumbnail medialive:DescribeInputSecurityGroup medialive:DescribeMultiplex medialive:DescribeMultiplexProgram medialive:DescribeNetwork medialive:DescribeNode medialive:DescribeOffering medialive:DescribeReservation medialive:DescribeSchedule medialive:DescribeSdiSource medialive:DescribeThumbnails medialive:GetCloudWatchAlarmTemplate medialive:GetCloudWatchAlarmTemplateGroup medialive:GetEventBridgeRuleTemplate medialive:GetEventBridgeRuleTemplateGroup medialive:GetSignalMap

服务前缀	操作
	medialive:ListAlerts medialive:ListChannelPlacementGroups medialive:ListChannels medialive:ListCloudWatchAlarmTemplateGroups medialive:ListCloudWatchAlarmTemplates medialive:ListClusterAlerts medialive:ListClusters medialive:ListEventBridgeRuleTemplateGroups medialive:ListEventBridgeRuleTemplates medialive:ListInputDeviceTransfers medialive:ListInputDevices medialive:ListInputSecurityGroups medialive:ListInputs medialive:ListMultiplexAlerts medialive:ListMultiplexPrograms medialive:ListMultiplexes medialive:ListNetworks medialive:ListNodes medialive:ListOfferings medialive:ListReservations medialive:ListSdiSources

服务前缀	操作
	medialive:ListSignalMaps medialive:ListVersions medialive:PurchaseOffering medialive:RebootInputDevice medialive:RejectInputDeviceTransfer medialive:RestartChannelPipelines medialive:StartChannel medialive:StartDeleteMonitorDeployment medialive:StartInputDevice medialive:StartInputDeviceMaintenanceWindow medialive:StartMonitorDeployment medialive:StartMultiplex medialive:StartUpdateSignalMap medialive:StopChannel medialive:StopInputDevice medialive:StopMultiplex medialive:TransferInputDevice medialive:UpdateAccountConfiguration medialive:UpdateChannel medialive:UpdateChannelClass medialive:UpdateChannelPlacementGroup

服务前缀	操作
	medialive:UpdateCloudWatchAlarmTemplate medialive:UpdateCloudWatchAlarmTemplateGroup medialive:UpdateCluster medialive:UpdateEventBridgeRuleTemplate medialive:UpdateEventBridgeRuleTemplateGroup medialive:UpdateInput medialive:UpdateInputDevice medialive:UpdateInputSecurityGroup medialive:UpdateMultiplex medialive:UpdateMultiplexProgram medialive:UpdateNetwork medialive:UpdateNode medialive:UpdateNodeState medialive:UpdateReservation medialive:UpdateSdiSource

服务前缀	操作
mediastore	mediastore:CreateContainer
	mediastore:DeleteContainer
	mediastore:DeleteContainerPolicy
	mediastore:DeleteCorsPolicy
	mediastore:DeleteLifecyclePolicy
	mediastore:DeleteMetricPolicy
	mediastore:DescribeContainer
	mediastore:GetContainerPolicy
	mediastore:GetCorsPolicy
	mediastore:GetLifecyclePolicy
	mediastore:GetMetricPolicy
	mediastore:ListContainers
	mediastore:PutContainerPolicy
	mediastore:PutCorsPolicy
	mediastore:PutLifecyclePolicy
	mediastore:PutMetricPolicy
	mediastore:StartAccessLogging
	mediastore:StopAccessLogging

服务前缀	操作
mediatailor	mediatailor:ConfigureLogsForPlaybackConfiguration mediatailor:CreateChannel mediatailor:CreateLiveSource mediatailor:CreatePrefetchSchedule mediatailor:CreateProgram mediatailor:CreateSourceLocation mediatailor:CreateVodSource mediatailor:DeleteChannel mediatailor:DeleteChannelPolicy mediatailor:DeleteLiveSource mediatailor:DeletePlaybackConfiguration mediatailor:DeletePrefetchSchedule mediatailor:DeleteProgram mediatailor:DeleteSourceLocation mediatailor:DeleteVodSource mediatailor:DescribeChannel mediatailor:DescribeLiveSource mediatailor:DescribeProgram mediatailor:DescribeSourceLocation mediatailor:DescribeVodSource mediatailor:GetChannelPolicy

服务前缀	操作
	mediatailor:GetChannelSchedule mediatailor:GetPlaybackConfiguration mediatailor:GetPrefetchSchedule mediatailor:ListAlerts mediatailor:ListChannels mediatailor:ListLiveSources mediatailor:ListPlaybackConfigurations mediatailor:ListPrefetchSchedules mediatailor:ListSourceLocations mediatailor:ListVodSources mediatailor:PutChannelPolicy mediatailor:PutPlaybackConfiguration mediatailor:StartChannel mediatailor:StopChannel mediatailor:UpdateChannel mediatailor:UpdateLiveSource mediatailor:UpdateProgram mediatailor:UpdateSourceLocation mediatailor:UpdateVodSource

服务前缀	操作
memorydb	memorydb:BatchUpdateCluster
	memorydb:CopySnapshot
	memorydb:CreateAcl
	memorydb:CreateCluster
	memorydb:CreateMultiRegionCluster
	memorydb:CreateParameterGroup
	memorydb:CreateSnapshot
	memorydb:CreateSubnetGroup
	memorydb:CreateUser
	memorydb:DeleteAcl
	memorydb:DeleteCluster
	memorydb:DeleteMultiRegionCluster
	memorydb:DeleteParameterGroup
	memorydb:DeleteSnapshot
	memorydb:DeleteSubnetGroup
	memorydb:DeleteUser
	memorydb:DescribeAcls
	memorydb:DescribeClusters
	memorydb:DescribeEngineVersions
	memorydb:DescribeEvents
	memorydb:DescribeMultiRegionClusters

服务前缀	操作
	memorydb:DescribeMultiRegionParameterGroups
	memorydb:DescribeMultiRegionParameters
	memorydb:DescribeParameterGroups
	memorydb:DescribeParameters
	memorydb:DescribeReservedNodes
	memorydb:DescribeReservedNodesOfferings
	memorydb:DescribeServiceUpdates
	memorydb:DescribeSnapshots
	memorydb:DescribeSubnetGroups
	memorydb:DescribeUsers
	memorydb:FailoverShard
	memorydb:ListAllowedMultiRegionClusterUpdates
	memorydb:ListAllowedNodeTypeUpdates
	memorydb:PurchaseReservedNodesOffering
	memorydb:ResetParameterGroup
	memorydb:UpdateAcl
	memorydb:UpdateCluster
	memorydb:UpdateMultiRegionCluster
	memorydb:UpdateParameterGroup
	memorydb:UpdateSubnetGroup
	memorydb:UpdateUser

服务前缀	操作
mgh	mgh:AssociateCreatedArtifact mgh:AssociateDiscoveredResource mgh:AssociateSourceResource mgh:CreateHomeRegionControl mgh:CreateProgressUpdateStream mgh:DeleteHomeRegionControl mgh:DeleteProgressUpdateStream mgh:DescribeApplicationState mgh:DescribeHomeRegionControls mgh:DescribeMigrationTask mgh:DisassociateCreatedArtifact mgh:DisassociateDiscoveredResource mgh:DisassociateSourceResource mgh:GetHomeRegion mgh:ImportMigrationTask mgh:ListApplicationStates mgh:ListCreatedArtifacts mgh:ListDiscoveredResources mgh:ListMigrationTaskUpdates mgh:ListMigrationTasks mgh:ListProgressUpdateStreams

服务前缀	操作
	mgh:ListSourceResources mgh:NotifyApplicationState mgh:NotifyMigrationTaskState mgh:PutResourceAttributes

服务前缀	操作
mgn	mgn:ArchiveApplication
	mgn:ArchiveWave
	mgn:AssociateApplications
	mgn:AssociateSourceServers
	mgn:ChangeServerLifeCycleState
	mgn:CreateApplication
	mgn:CreateConnector
	mgn:CreateLaunchConfigurationTemplate
	mgn:CreateReplicationConfigurationTemplate
	mgn:CreateWave
	mgn:DeleteApplication
	mgn:DeleteConnector
	mgn:DeleteJob
	mgn:DeleteLaunchConfigurationTemplate
	mgn:DeleteReplicationConfigurationTemplate
	mgn:DeleteSourceServer
	mgn:DeleteVcenterClient
	mgn:DeleteWave
	mgn:DescribeJobLogItems
	mgn:DescribeJobs
	mgn:DescribeLaunchConfigurationTemplates

服务前缀	操作
	mgn:DescribeReplicationConfigurationTemplates
	mgn:DescribeVcenterClients
	mgn:DisassociateApplications
	mgn:DisassociateSourceServers
	mgn:DisconnectFromService
	mgn:FinalizeCutover
	mgn:GetReplicationConfiguration
	mgn:InitializeService
	mgn:ListConnectors
	mgn:ListExportErrors
	mgn:ListExports
	mgn:ListImportErrors
	mgn:ListImports
	mgn:ListManagedAccounts
	mgn:ListSourceServerActions
	mgn:ListTemplateActions
	mgn:MarkAsArchived
	mgn:PauseReplication
	mgn:PutSourceServerAction
	mgn:PutTemplateAction
	mgn:RemoveSourceServerAction

服务前缀	操作
	mgn:RemoveTemplateAction
	mgn:ResumeReplication
	mgn:RetryDataReplication
	mgn:StartCutover
	mgn:StartExport
	mgn:StartImport
	mgn:StartReplication
	mgn:StartTest
	mgn:StopReplication
	mgn:TerminateTargetInstances
	mgn:UnarchiveApplication
	mgn:UnarchiveWave
	mgn:UpdateApplication
	mgn:UpdateConnector
	mgn:UpdateLaunchConfigurationTemplate
	mgn:UpdateReplicationConfiguration
	mgn:UpdateReplicationConfigurationTemplate
	mgn:UpdateSourceServer
	mgn:UpdateSourceServerReplicationType
	mgn:UpdateWave

服务前缀	操作
migrationhub-strategy	migrationhub-strategy:GetAntiPattern migrationhub-strategy:GetApplicationComponentDetails migrationhub-strategy:GetApplicationComponentStrategies migrationhub-strategy:GetAssessment migrationhub-strategy:GetImportFileTask migrationhub-strategy:GetLatestAssessmentId migrationhub-strategy:GetMessage migrationhub-strategy:GetPortfolioPreferences migrationhub-strategy:GetPortfolioSummary migrationhub-strategy:GetRecommendationReportDetails migrationhub-strategy:GetServerDetails migrationhub-strategy:GetServerStrategies migrationhub-strategy:ListAnalyzableServers migrationhub-strategy:ListAntiPatterns migrationhub-strategy:ListApplicationComponents migrationhub-strategy:ListCollectors migrationhub-strategy:ListImportFileTask migrationhub-strategy:ListJarArtifacts migrationhub-strategy:ListServers migrationhub-strategy:PutLogData migrationhub-strategy:PutMetricData

服务前缀	操作
	migrationhub-strategy:PutPortfolioPreferences migrationhub-strategy:RegisterCollector migrationhub-strategy:SendMessage migrationhub-strategy:StartAssessment migrationhub-strategy:StartImportFileTask migrationhub-strategy:StartRecommendationReportGeneration migrationhub-strategy:StopAssessment migrationhub-strategy:UpdateApplicationComponentConfig migrationhub-strategy:UpdateCollectorConfiguration migrationhub-strategy:UpdateServerConfig

服务前缀	操作
mobiletargeting	mobiletargeting:CreateApp
	mobiletargeting:CreateCampaign
	mobiletargeting:CreateEmailTemplate
	mobiletargeting:CreateExportJob
	mobiletargeting:CreateImportJob
	mobiletargeting:CreateInAppTemplate
	mobiletargeting:CreateJourney
	mobiletargeting:CreatePushTemplate
	mobiletargeting:CreateRecommenderConfiguration
	mobiletargeting:CreateSegment
	mobiletargeting:CreateSmsTemplate
	mobiletargeting:CreateVoiceTemplate
	mobiletargeting:DeleteAdmChannel
	mobiletargeting:DeleteApnsChannel
	mobiletargeting:DeleteApnsSandboxChannel
	mobiletargeting:DeleteApnsVoipChannel
	mobiletargeting:DeleteApnsVoipSandboxChannel
	mobiletargeting:DeleteApp
	mobiletargeting:DeleteBaiduChannel
	mobiletargeting:DeleteCampaign
	mobiletargeting:DeleteEmailChannel

服务前缀	操作
	mobiletargeting:DeleteEmailTemplate
	mobiletargeting:DeleteEndpoint
	mobiletargeting:DeleteEventStream
	mobiletargeting:DeleteGcmChannel
	mobiletargeting:DeleteInAppTemplate
	mobiletargeting:DeleteJourney
	mobiletargeting:DeletePushTemplate
	mobiletargeting:DeleteRecommenderConfiguration
	mobiletargeting:DeleteSegment
	mobiletargeting:DeleteSmsChannel
	mobiletargeting:DeleteSmsTemplate
	mobiletargeting:DeleteUserEndpoints
	mobiletargeting:DeleteVoiceChannel
	mobiletargeting:DeleteVoiceTemplate
	mobiletargeting:GetAdmChannel
	mobiletargeting:GetApnsChannel
	mobiletargeting:GetApnsSandboxChannel
	mobiletargeting:GetApnsVoipChannel
	mobiletargeting:GetApnsVoipSandboxChannel
	mobiletargeting:GetApp
	mobiletargeting:GetApplicationDateRangeKpi

服务前缀	操作
	mobiletargeting:GetApplicationSettings
	mobiletargeting:GetApps
	mobiletargeting:GetBaiduChannel
	mobiletargeting:GetCampaign
	mobiletargeting:GetCampaignActivities
	mobiletargeting:GetCampaignDateRangeKpi
	mobiletargeting:GetCampaignVersion
	mobiletargeting:GetCampaignVersions
	mobiletargeting:GetCampaigns
	mobiletargeting:GetChannels
	mobiletargeting:GetEmailChannel
	mobiletargeting:GetEmailTemplate
	mobiletargeting:GetEndpoint
	mobiletargeting:GetEventStream
	mobiletargeting:GetExportJob
	mobiletargeting:GetExportJobs
	mobiletargeting:GetGcmChannel
	mobiletargeting:GetImportJob
	mobiletargeting:GetImportJobs
	mobiletargeting:GetInAppMessages
	mobiletargeting:GetInAppTemplate

服务前缀	操作
	mobiletargeting:GetJourney
	mobiletargeting:GetJourneyDateRangeKpi
	mobiletargeting:GetJourneyExecutionActivityMetrics
	mobiletargeting:GetJourneyExecutionMetrics
	mobiletargeting:GetJourneyRunExecutionActivityMetrics
	mobiletargeting:GetJourneyRunExecutionMetrics
	mobiletargeting:GetJourneyRuns
	mobiletargeting:GetPushTemplate
	mobiletargeting:GetRecommenderConfiguration
	mobiletargeting:GetRecommenderConfigurations
	mobiletargeting:GetSegment
	mobiletargeting:GetSegmentExportJobs
	mobiletargeting:GetSegmentImportJobs
	mobiletargeting:GetSegmentVersion
	mobiletargeting:GetSegmentVersions
	mobiletargeting:GetSegments
	mobiletargeting:GetSmsChannel
	mobiletargeting:GetSmsTemplate
	mobiletargeting:GetUserEndpoints
	mobiletargeting:GetVoiceChannel
	mobiletargeting:GetVoiceTemplate

服务前缀	操作
	mobiletargeting:ListJourneys
	mobiletargeting:ListTemplateVersions
	mobiletargeting:ListTemplates
	mobiletargeting:PhoneNumberValidate
	mobiletargeting:PutEventStream
	mobiletargeting:RemoveAttributes
	mobiletargeting:UpdateAdmChannel
	mobiletargeting:UpdateApnsChannel
	mobiletargeting:UpdateApnsSandboxChannel
	mobiletargeting:UpdateApnsVoipChannel
	mobiletargeting:UpdateApnsVoipSandboxChannel
	mobiletargeting:UpdateApplicationSettings
	mobiletargeting:UpdateBaiduChannel
	mobiletargeting:UpdateCampaign
	mobiletargeting:UpdateEmailChannel
	mobiletargeting:UpdateEmailTemplate
	mobiletargeting:UpdateEndpoint
	mobiletargeting:UpdateEndpointsBatch
	mobiletargeting:UpdateGcmChannel
	mobiletargeting:UpdateInAppTemplate
	mobiletargeting:UpdateJourney

服务前缀	操作
	mobiletargeting:UpdateJourneyState
	mobiletargeting:UpdatePushTemplate
	mobiletargeting:UpdateRecommenderConfiguration
	mobiletargeting:UpdateSegment
	mobiletargeting:UpdateSmsChannel
	mobiletargeting:UpdateSmsTemplate
	mobiletargeting:UpdateTemplateActiveVersion
	mobiletargeting:UpdateVoiceChannel
	mobiletargeting:UpdateVoiceTemplate
	mobiletargeting:VerifyOTPMessage

服务前缀	操作
mq	mq:CreateBroker mq:CreateConfiguration mq:CreateUser mq:DeleteBroker mq:DeleteConfiguration mq:DeleteUser mq:DescribeBroker mq:DescribeBrokerEngineTypes mq:DescribeBrokerInstanceOptions mq:DescribeConfiguration mq:DescribeConfigurationRevision mq:DescribeUser mq:ListBrokers mq:ListConfigurationRevisions mq:ListConfigurations mq:ListUsers mq:Promote mq:RebootBroker mq:UpdateBroker mq:UpdateConfiguration mq:UpdateUser

服务前缀	操作
networkmanager	networkmanager:AcceptAttachment
	networkmanager:AssociateConnectPeer
	networkmanager:AssociateCustomerGateway
	networkmanager:AssociateLink
	networkmanager:AssociateTransitGatewayConnectPeer
	networkmanager:CreateConnectAttachment
	networkmanager:CreateConnectPeer
	networkmanager:CreateConnection
	networkmanager:CreateCoreNetwork
	networkmanager:CreateDevice
	networkmanager:CreateDirectConnectGatewayAttachment
	networkmanager:CreateGlobalNetwork
	networkmanager:CreateLink
	networkmanager:CreateSite
	networkmanager:CreateSiteToSiteVpnAttachment
	networkmanager:CreateTransitGatewayPeering
	networkmanager:CreateTransitGatewayRouteTableAttachment
	networkmanager:CreateVpcAttachment
	networkmanager:DeleteAttachment
	networkmanager:DeleteConnectPeer
	networkmanager:DeleteConnection

服务前缀	操作
	networkmanager:DeleteCoreNetwork
	networkmanager:DeleteCoreNetworkPolicyVersion
	networkmanager:DeleteDevice
	networkmanager:DeleteGlobalNetwork
	networkmanager:DeleteLink
	networkmanager:DeletePeering
	networkmanager:DeleteResourcePolicy
	networkmanager:DeleteSite
	networkmanager:DeregisterTransitGateway
	networkmanager:DescribeGlobalNetworks
	networkmanager:DisassociateConnectPeer
	networkmanager:DisassociateCustomerGateway
	networkmanager:DisassociateLink
	networkmanager:DisassociateTransitGatewayConnectPeer
	networkmanager:ExecuteCoreNetworkChangeSet
	networkmanager:GetConnectAttachment
	networkmanager:GetConnectPeer
	networkmanager:GetConnectPeerAssociations
	networkmanager:GetConnections
	networkmanager:GetCoreNetwork
	networkmanager:GetCoreNetworkChangeEvents

服务前缀	操作
	networkmanager:GetCoreNetworkChangeSet
	networkmanager:GetCoreNetworkPolicy
	networkmanager:GetCustomerGatewayAssociations
	networkmanager:GetDevices
	networkmanager:GetLinkAssociations
	networkmanager:GetLinks
	networkmanager:GetNetworkResourceCounts
	networkmanager:GetNetworkResourceRelationships
	networkmanager:GetNetworkResources
	networkmanager:GetNetworkRoutes
	networkmanager:GetNetworkTelemetry
	networkmanager:GetResourcePolicy
	networkmanager:GetRouteAnalysis
	networkmanager:GetSiteToSiteVpnAttachment
	networkmanager:GetSites
	networkmanager:GetTransitGatewayConnectPeerAssociations
	networkmanager:GetTransitGatewayPeering
	networkmanager:GetTransitGatewayRegistrations
	networkmanager:GetTransitGatewayRouteTableAttachment
	networkmanager:GetVpcAttachment
	networkmanager:ListAttachmentRoutingPolicyAssociations

服务前缀	操作
	networkmanager:ListAttachments
	networkmanager:ListConnectPeers
	networkmanager:ListCoreNetworkPolicyVersions
	networkmanager:ListCoreNetworkPrefixListAssociations
	networkmanager:ListCoreNetworkRoutingInformation
	networkmanager:ListCoreNetworks
	networkmanager:ListOrganizationServiceAccessStatus
	networkmanager:ListPeerings
	networkmanager:PutAttachmentRoutingPolicyLabel
	networkmanager:PutCoreNetworkPolicy
	networkmanager:PutResourcePolicy
	networkmanager:RegisterTransitGateway
	networkmanager:RejectAttachment
	networkmanager:RemoveAttachmentRoutingPolicyLabel
	networkmanager:RestoreCoreNetworkPolicyVersion
	networkmanager:StartOrganizationServiceAccessUpdate
	networkmanager:StartRouteAnalysis
	networkmanager:UpdateConnection
	networkmanager:UpdateCoreNetwork
	networkmanager:UpdateDevice
	networkmanager:UpdateDirectConnectGatewayAttachment

服务前缀	操作
	networkmanager:UpdateGlobalNetwork networkmanager:UpdateLink networkmanager:UpdateNetworkResourceMetadata networkmanager:UpdateSite networkmanager:UpdateVpcAttachment

服务前缀	操作
nimble	nimble:AcceptEulas nimble:CreateLaunchProfile nimble:CreateStreamingImage nimble:CreateStreamingSession nimble:CreateStreamingSessionStream nimble:CreateStudio nimble:CreateStudioComponent nimble:DeleteLaunchProfile nimble:DeleteLaunchProfileMember nimble:DeleteStreamingImage nimble:DeleteStreamingSession nimble:DeleteStudio nimble:DeleteStudioComponent nimble:DeleteStudioMember nimble:GetEula nimble:GetLaunchProfileDetails nimble:GetStreamingImage nimble:GetStreamingSession nimble:GetStreamingSessionBackup nimble:GetStreamingSessionStream nimble:GetStudio

服务前缀	操作
	nimble:GetStudioComponent nimble:GetStudioMember nimble:ListEulas nimble:ListLaunchProfileMembers nimble:ListLaunchProfiles nimble:ListStreamingImages nimble:ListStreamingSessionBackups nimble:ListStreamingSessions nimble:ListStudioComponents nimble:ListStudioMembers nimble:ListStudios nimble:PutLaunchProfileMembers nimble:PutStudioMembers nimble:StartStreamingSession nimble:StartStudioSSOConfigurationRepair nimble:StopStreamingSession nimble:UpdateLaunchProfile nimble:UpdateLaunchProfileMember nimble:UpdateStreamingImage nimble:UpdateStudio nimble:UpdateStudioComponent

服务前缀	操作
omics	omics:AbortMultipartReadSetUpload
	omics:AcceptShare
	omics:BatchDeleteReadSet
	omics:CancelAnnotationImportJob
	omics:CancelRun
	omics:CancelVariantImportJob
	omics:CompleteMultipartReadSetUpload
	omics:CreateAnnotationStore
	omics:CreateAnnotationStoreVersion
	omics:CreateMultipartReadSetUpload
	omics:CreateReferenceStore
	omics:CreateRunGroup
	omics:CreateSequenceStore
	omics:CreateShare
	omics:CreateVariantStore
	omics:CreateWorkflow
	omics:CreateWorkflowVersion
	omics:DeleteAnnotationStore
	omics:DeleteAnnotationStoreVersions
	omics:DeleteReference
	omics:DeleteReferenceStore

服务前缀	操作
	omics:DeleteRun
	omics:DeleteRunGroup
	omics:DeleteSequenceStore
	omics:DeleteShare
	omics:DeleteVariantStore
	omics:DeleteWorkflow
	omics:DeleteWorkflowVersion
	omics:GetAnnotationImportJob
	omics:GetAnnotationStore
	omics:GetAnnotationStoreVersion
	omics:GetReadSet
	omics:GetReadSetActivationJob
	omics:GetReadSetExportJob
	omics:GetReadSetImportJob
	omics:GetReadSetMetadata
	omics:GetReference
	omics:GetReferenceImportJob
	omics:GetReferenceMetadata
	omics:GetReferenceStore
	omics:GetRun
	omics:GetRunGroup

服务前缀	操作
	omics:GetRunTask
	omics:GetSequenceStore
	omics:GetShare
	omics:GetVariantImportJob
	omics:GetVariantStore
	omics:GetWorkflow
	omics:GetWorkflowVersion
	omics:ListAnnotationImportJobs
	omics:ListAnnotationStoreVersions
	omics:ListAnnotationStores
	omics:ListMultipartReadSetUploads
	omics:ListReadSetActivationJobs
	omics:ListReadSetExportJobs
	omics:ListReadSetImportJobs
	omics:ListReadSetUploadParts
	omics:ListReadSets
	omics:ListReferenceImportJobs
	omics:ListReferenceStores
	omics:ListReferences
	omics:ListRunGroups
	omics:ListRunTasks

服务前缀	操作
	omics:ListRuns
	omics:ListSequenceStores
	omics:ListShares
	omics:ListVariantImportJobs
	omics:ListVariantStores
	omics:ListWorkflowVersions
	omics:ListWorkflows
	omics:StartAnnotationImportJob
	omics:StartReadSetActivationJob
	omics:StartReadSetExportJob
	omics:StartReadSetImportJob
	omics:StartReferenceImportJob
	omics:StartRun
	omics:StartVariantImportJob
	omics:UpdateAnnotationStore
	omics:UpdateAnnotationStoreVersion
	omics:UpdateRunGroup
	omics:UpdateVariantStore
	omics:UpdateWorkflow
	omics:UpdateWorkflowVersion
	omics:UploadReadSetPart

服务前缀	操作
opsworks	opsworks:AssignInstance opsworks:AssignVolume opsworks:AssociateElasticIp opsworks:AttachElasticLoadBalancer opsworks:CloneStack opsworks:CreateApp opsworks:CreateDeployment opsworks:CreateInstance opsworks:CreateLayer opsworks:CreateStack opsworks:CreateUserProfile opsworks:DeleteApp opsworks:DeleteInstance opsworks:DeleteLayer opsworks:DeleteStack opsworks:DeleteUserProfile opsworks:DeregisterEcsCluster opsworks:DeregisterElasticIp opsworks:DeregisterInstance opsworks:DeregisterRdsDbInstance opsworks:DeregisterVolume

服务前缀	操作
	opsworks:DescribeAgentVersions opsworks:DescribeApps opsworks:DescribeCommands opsworks:DescribeDeployments opsworks:DescribeEcsClusters opsworks:DescribeElasticIps opsworks:DescribeElasticLoadBalancers opsworks:DescribeInstances opsworks:DescribeLayers opsworks:DescribeLoadBasedAutoScaling opsworks:DescribeMyUserProfile opsworks:DescribeOperatingSystems opsworks:DescribePermissions opsworks:DescribeRaidArrays opsworks:DescribeRdsDbInstances opsworks:DescribeServiceErrors opsworks:DescribeStackProvisioningParameters opsworks:DescribeStackSummary opsworks:DescribeStacks opsworks:DescribeTimeBasedAutoScaling opsworks:DescribeUserProfiles

服务前缀	操作
	opsworks:DescribeVolumes opsworks:DetachElasticLoadBalancer opsworks:DisassociateElasticIp opsworks:GetHostnameSuggestion opsworks:GrantAccess opsworks:RebootInstance opsworks:RegisterEcsCluster opsworks:RegisterElasticIp opsworks:RegisterInstance opsworks:RegisterRdsDbInstance opsworks:RegisterVolume opsworks:SetLoadBasedAutoScaling opsworks:SetPermission opsworks:SetTimeBasedAutoScaling opsworks:StartInstance opsworks:StartStack opsworks:StopInstance opsworks:StopStack opsworks:UnassignInstance opsworks:UnassignVolume opsworks:UpdateApp

服务前缀	操作
	opsworks:UpdateElasticIp opsworks:UpdateInstance opsworks:UpdateLayer opsworks:UpdateMyUserProfile opsworks:UpdateRdsDbInstance opsworks:UpdateStack opsworks:UpdateUserProfile opsworks:UpdateVolume

服务前缀	操作
opsworks-cm	opsworks-cm:AssociateNode
	opsworks-cm:CreateBackup
	opsworks-cm:CreateServer
	opsworks-cm:DeleteBackup
	opsworks-cm:DeleteServer
	opsworks-cm:DescribeAccountAttributes
	opsworks-cm:DescribeBackups
	opsworks-cm:DescribeEvents
	opsworks-cm:DescribeNodeAssociationStatus
	opsworks-cm:DescribeServers
	opsworks-cm:DisassociateNode
	opsworks-cm:ExportServerEngineAttribute
	opsworks-cm:RestoreServer
	opsworks-cm:StartMaintenance
	opsworks-cm:UpdateServer
	opsworks-cm:UpdateServerEngineAttributes

服务前缀	操作
组织	organizations:AcceptHandshake organizations:AttachPolicy organizations:CancelHandshake organizations:CloseAccount organizations:CreateAccount organizations:CreateGovCloudAccount organizations:CreateOrganization organizations:CreateOrganizationalUnit organizations:CreatePolicy organizations:DeclineHandshake organizations:DeleteOrganization organizations:DeleteOrganizationalUnit organizations:DeletePolicy organizations:DeleteResourcePolicy organizations:DeregisterDelegatedAdministrator organizations:DescribeAccount organizations:DescribeCreateAccountStatus organizations:DescribeEffectivePolicy organizations:DescribeHandshake organizations:DescribeOrganization organizations:DescribeOrganizationalUnit

服务前缀	操作
	organizations:DescribePolicy organizations:DescribeResourcePolicy organizations:DescribeResponsibilityTransfer organizations:DetachPolicy organizations:DisableAWSServiceAccess organizations:DisablePolicyType organizations:EnableAWSServiceAccess organizations:EnableAllFeatures organizations:EnablePolicyType organizations:InviteAccountToOrganization organizations:LeaveOrganization organizations:ListAWSServiceAccessForOrganization organizations:ListAccounts organizations:ListAccountsForParent organizations:ListAccountsWithInvalidEffectivePolicy organizations:ListChildren organizations:ListCreateAccountStatus organizations:ListDelegatedAdministrators organizations:ListDelegatedServicesForAccount organizations:ListHandshakesForAccount organizations:ListHandshakesForOrganization

服务前缀	操作
	organizations:ListInboundResponsibilityTransfers organizations:ListOrganizationalUnitsForParent organizations:ListOutboundResponsibilityTransfers organizations:ListParents organizations:ListPolicies organizations:ListPoliciesForTarget organizations:ListRoots organizations:ListTargetsForPolicy organizations:MoveAccount organizations:PutResourcePolicy organizations:RegisterDelegatedAdministrator organizations:RemoveAccountFromOrganization organizations:TerminateResponsibilityTransfer organizations:UpdateOrganizationalUnit organizations:UpdatePolicy organizations:UpdateResponsibilityTransfer

服务前缀	操作
outposts	outposts:CancelCapacityTask
	outposts:CancelOrder
	outposts:CreateOrder
	outposts:CreateOutpost
	outposts:CreatePrivateConnectivityConfig
	outposts:CreateSite
	outposts:DeleteOutpost
	outposts:DeleteSite
	outposts:GetCapacityTask
	outposts:GetCatalogItem
	outposts:GetConnection
	outposts:GetOrder
	outposts:GetOutpost
	outposts:GetOutpostBillingInformation
	outposts:GetOutpostInstanceTypes
	outposts:GetOutpostSupportedInstanceTypes
	outposts:GetPrivateConnectivityConfig
	outposts:GetSite
	outposts:GetSiteAddress
	outposts:ListAssetInstances
	outposts:ListAssets

服务前缀	操作
	outposts:ListBlockingInstancesForCapacityTask
	outposts:ListCapacityTasks
	outposts:ListCatalogItems
	outposts:ListOrders
	outposts:ListOutposts
	outposts:ListSites
	outposts:StartCapacityTask
	outposts:StartConnection
	outposts:UpdateOutpost
	outposts:UpdateSite
	outposts:UpdateSiteAddress
	outposts:UpdateSiteRackPhysicalProperties

服务前缀	操作
panorama	panorama:CreateApplicationInstance panorama:CreateJobForDevices panorama:CreateNodeFromTemplateJob panorama:CreatePackage panorama:CreatePackageImportJob panorama:DeleteDevice panorama:DeletePackage panorama:DeregisterPackageVersion panorama:DescribeApplicationInstance panorama:DescribeApplicationInstanceDetails panorama:DescribeDevice panorama:DescribeDeviceJob panorama:DescribeNode panorama:DescribeNodeFromTemplateJob panorama:DescribePackage panorama:DescribePackageImportJob panorama:DescribePackageVersion panorama:ListApplicationInstanceDependencies panorama:ListApplicationInstanceNodeInstances panorama:ListApplicationInstances panorama:ListDevices

服务前缀	操作
	panorama:ListDevicesJobs panorama:ListNodeFromTemplateJobs panorama:ListNodes panorama:ListPackageImportJobs panorama:ListPackages panorama:ProvisionDevice panorama:RegisterPackageVersion panorama:RemoveApplicationInstance panorama:SignalApplicationInstanceNodeInstances panorama:UpdateDeviceMetadata
pi	pi:CreatePerformanceAnalysisReport pi:DeletePerformanceAnalysisReport pi:DescribeDimensionKeys pi:GetDimensionKeyDetails pi:GetPerformanceAnalysisReport pi:GetResourceMetadata pi:GetResourceMetrics pi:ListAvailableResourceDimensions pi:ListAvailableResourceMetrics pi:ListPerformanceAnalysisReports

服务前缀	操作
pipes	pipes:CreatePipe pipes:DeletePipe pipes:DescribePipe pipes:ListPipes pipes:StartPipe pipes:StopPipe pipes:UpdatePipe
polly	polly:DeleteLexicon polly:DescribeVoices polly:GetLexicon polly:GetSpeechSynthesisTask polly:ListLexicons polly:ListSpeechSynthesisTasks polly:PutLexicon polly:StartSpeechSynthesisTask polly:SynthesizeSpeech

服务前缀	操作
配置文件	profile:AddProfileKey profile:BatchGetCalculatedAttributeForProfile profile:BatchGetProfile profile:CreateCalculatedAttributeDefinition profile:CreateDomain profile:CreateEventStream profile:CreateProfile profile:CreateRecommender profile:CreateSegmentDefinition profile:CreateSegmentEstimate profile:CreateSegmentSnapshot profile:CreateUploadJob profile:DeleteCalculatedAttributeDefinition profile:DeleteDomain profile:DeleteDomainObjectType profile:DeleteEventStream profile:DeleteIntegration profile:DeleteProfile profile:DeleteProfileKey profile:DeleteProfileObject profile:DeleteProfileObjectType

服务前缀	操作
	profile:DeleteRecommender profile:DeleteSegmentDefinition profile:DeleteWorkflow profile:DetectProfileObjectType profile:GetAutoMergingPreview profile:GetCalculatedAttributeDefinition profile:GetCalculatedAttributeForProfile profile:GetDomain profile:GetDomainObjectType profile:GetEventStream profile:GetIdentityResolutionJob profile:GetIntegration profile:GetMatches profile:GetObjectTypeAttributeStatistics profile:GetProfileObjectType profile:GetProfileObjectTypeTemplate profile:GetProfileRecommendations profile:GetRecommender profile:GetSegmentDefinition profile:GetSegmentEstimate profile:GetSegmentMembership

服务前缀	操作
	profile:GetSegmentSnapshot profile:GetSimilarProfiles profile:GetUploadJob profile:GetUploadJobPath profile:GetWorkflow profile:GetWorkflowSteps profile:ListAccountIntegrations profile:ListCalculatedAttributeDefinitions profile:ListCalculatedAttributesForProfile profile:ListDomainLayouts profile:ListDomainObjectTypes profile:ListDomains profile:ListEventStreams profile:ListIdentityResolutionJobs profile:ListIntegrations profile:ListObjectTypeAttributeValues profile:ListObjectTypeAttributes profile:ListProfileAttributeValues profile:ListProfileObjectTypeTemplates profile:ListProfileObjectTypes profile:ListProfileObjects

服务前缀	操作
	profile:ListRecommenderRecipes profile:ListRecommenders profile:ListRuleBasedMatches profile:ListSegmentDefinitions profile:ListUploadJobs profile:ListWorkflows profile:MergeProfiles profile:PutDomainObjectType profile:PutIntegration profile:PutProfileObject profile:PutProfileObjectType profile:SearchProfiles profile:StartRecommender profile:StartUploadJob profile:StopRecommender profile:StopUploadJob profile:UpdateCalculatedAttributeDefinition profile:UpdateDomain profile:UpdateProfile profile:UpdateRecommender

服务前缀	操作
qldb	qldb:CancelJournalKinesisStream
	qldb:CreateLedger
	qldb:DeleteLedger
	qldb:DescribeJournalKinesisStream
	qldb:DescribeJournalS3Export
	qldb:DescribeLedger
	qldb:ExportJournalToS3
	qldb:GetBlock
	qldb:GetDigest
	qldb:GetRevision
	qldb:ListJournalKinesisStreamsForLedger
	qldb:ListJournalS3Exports
	qldb:ListJournalS3ExportsForLedger
	qldb:ListLedgers
	qldb:StreamJournalToKinesis
	qldb:UpdateLedger
	qldb:UpdateLedgerPermissionsMode

服务前缀	操作
ram	ram:AcceptResourceShareInvitation ram:AssociateResourceShare ram:AssociateResourceSharePermission ram:CreatePermission ram:CreatePermissionVersion ram:CreateResourceShare ram:DeletePermission ram:DeletePermissionVersion ram:DeleteResourceShare ram:DisassociateResourceShare ram:DisassociateResourceSharePermission ram:EnableSharingWithAwsOrganization ram:GetPermission ram:GetResourcePolicies ram:GetResourceShareAssociations ram:GetResourceShareInvitations ram:GetResourceShares ram:ListPendingInvitationResources ram:ListPermissionAssociations ram:ListPermissionVersions ram:ListPermissions

服务前缀	操作
	ram:ListPrincipals ram:ListReplacePermissionAssociationsWork ram:ListResourceSharePermissions ram:ListResourceTypes ram:ListResources ram:PromotePermissionCreatedFromPolicy ram:PromoteResourceShareCreatedFromPolicy ram:RejectResourceShareInvitation ram:ReplacePermissionAssociations ram:SetDefaultPermissionVersion ram:UpdateResourceShare
rbin	rbin:CreateRule rbin:DeleteRule rbin:GetRule rbin:ListRules rbin:LockRule rbin:UnlockRule rbin:UpdateRule

服务前缀	操作
rds	rds:AddRoleToDBCluster rds:AddRoleToDBInstance rds:AddSourceIdentifierToSubscription rds:ApplyPendingMaintenanceAction rds:AuthorizeDBSecurityGroupIngress rds:BacktrackDBCluster rds:CancelExportTask rds:CopyDBClusterParameterGroup rds:CopyDBClusterSnapshot rds:CopyDBParameterGroup rds:CopyDBSnapshot rds:CopyOptionGroup rds:CreateCustomDBEngineVersion rds:CreateDBClusterParameterGroup rds:CreateDBParameterGroup rds:CreateDBProxy rds:CreateDBProxyEndpoint rds:CreateDBSecurityGroup rds:CreateDBSubnetGroup rds:CreateEventSubscription rds:CreateGlobalCluster

服务前缀	操作
	rds:CreateOptionGroup rds:DeleteBlueGreenDeployment rds:DeleteDBClusterAutomatedBackup rds:DeleteDBClusterParameterGroup rds:DeleteDBClusterSnapshot rds:DeleteDBInstanceAutomatedBackup rds:DeleteDBParameterGroup rds:DeleteDBProxy rds:DeleteDBProxyEndpoint rds:DeleteDBSecurityGroup rds:DeleteDBSnapshot rds:DeleteDBSubnetGroup rds:DeleteEventSubscription rds:DeleteGlobalCluster rds:DeleteOptionGroup rds:DeregisterDBProxyTargets rds:DescribeAccountAttributes rds:DescribeBlueGreenDeployments rds:DescribeCertificates rds:DescribeDBClusterAutomatedBackups rds:DescribeDBClusterBacktracks

服务前缀	操作
	rds:DescribeDBClusterEndpoints rds:DescribeDBClusterParameterGroups rds:DescribeDBClusterParameters rds:DescribeDBClusterSnapshotAttributes rds:DescribeDBClusterSnapshots rds:DescribeDBClusters rds:DescribeDBEngineVersions rds:DescribeDBInstanceAutomatedBackups rds:DescribeDBInstances rds:DescribeDBLogFiles rds:DescribeDBMajorEngineVersions rds:DescribeDBParameterGroups rds:DescribeDBParameters rds:DescribeDBProxies rds:DescribeDBProxyEndpoints rds:DescribeDBProxyTargetGroups rds:DescribeDBProxyTargets rds:DescribeDBRecommendations rds:DescribeDBSecurityGroups rds:DescribeDBSnapshotAttributes rds:DescribeDBSnapshotTenantDatabases

服务前缀	操作
	rds:DescribeDBSnapshots rds:DescribeDBSubnetGroups rds:DescribeEngineDefaultClusterParameters rds:DescribeEngineDefaultParameters rds:DescribeEventCategories rds:DescribeEventSubscriptions rds:DescribeEvents rds:DescribeExportTasks rds:DescribeGlobalClusters rds:DescribeIntegrations rds:DescribeOptionGroupOptions rds:DescribeOptionGroups rds:DescribeOrderableDBInstanceOptions rds:DescribePendingMaintenanceActions rds:DescribeReservedDBInstances rds:DescribeReservedDBInstancesOfferings rds:DescribeSourceRegions rds:DescribeTenantDatabases rds:DescribeValidDBInstanceModifications rds:DownloadCompleteDBLogFile rds:DownloadDBLogFilePortion

服务前缀	操作
	rds:FailoverDBCluster rds:FailoverGlobalCluster rds:ModifyActivityStream rds:ModifyCertificates rds:ModifyCurrentDBClusterCapacity rds:ModifyDBClusterEndpoint rds:ModifyDBClusterParameterGroup rds:ModifyDBClusterSnapshotAttribute rds:ModifyDBParameterGroup rds:ModifyDBProxy rds:ModifyDBProxyEndpoint rds:ModifyDBProxyTargetGroup rds:ModifyDBRecommendation rds:ModifyDBSnapshot rds:ModifyDBSnapshotAttribute rds:ModifyDBSubnetGroup rds:ModifyEventSubscription rds:ModifyGlobalCluster rds:ModifyOptionGroup rds:ModifyTenantDatabase rds:PurchaseReservedDBInstancesOffering

服务前缀	操作
	rds:RebootDBCluster rds:RegisterDBProxyTargets rds:RemoveFromGlobalCluster rds:RemoveRoleFromDBCluster rds:RemoveRoleFromDBInstance rds:RemoveSourceIdentifierFromSubscription rds:ResetDBClusterParameterGroup rds:ResetDBParameterGroup rds:RestoreDBClusterFromS3 rds:RestoreDBClusterFromSnapshot rds:RestoreDBClusterToPointInTime rds:RestoreDBInstanceFromDBSnapshot rds:RestoreDBInstanceFromS3 rds:RestoreDBInstanceToPointInTime rds:RevokeDBSecurityGroupIngress rds:StartActivityStream rds:StartDBCluster rds:StartDBInstance rds:StartDBInstanceAutomatedBackupsReplication rds:StartExportTask rds:StopActivityStream

服务前缀	操作
	rds:StopDBCluster rds:StopDBInstance rds:StopDBInstanceAutomatedBackupsReplication rds:SwitchoverBlueGreenDeployment rds:SwitchoverGlobalCluster rds:SwitchoverReadReplica

服务前缀	操作
redshift	redshift:AcceptReservedNodeExchange
	redshift:AddPartner
	redshift:AssociateDataShareConsumer
	redshift:AuthorizeClusterSecurityGroupIngress
	redshift:AuthorizeDataShare
	redshift:AuthorizeEndpointAccess
	redshift:AuthorizeSnapshotAccess
	redshift:BatchDeleteClusterSnapshots
	redshift:BatchModifyClusterSnapshots
	redshift:CancelResize
	redshift:CopyClusterSnapshot
	redshift:CreateAuthenticationProfile
	redshift:CreateCluster
	redshift:CreateClusterParameterGroup
	redshift:CreateClusterSecurityGroup
	redshift:CreateClusterSnapshot
	redshift:CreateClusterSubnetGroup
	redshift:CreateCustomDomainAssociation
	redshift:CreateEndpointAccess
	redshift:CreateEventSubscription
	redshift:CreateHsmClientCertificate

服务前缀	操作
	redshift:CreateHsmConfiguration
	redshift:CreateIntegration
	redshift:CreateRedshiftIdcApplication
	redshift:CreateScheduledAction
	redshift:CreateSnapshotCopyGrant
	redshift:CreateSnapshotSchedule
	redshift:CreateUsageLimit
	redshift:DeauthorizeDataShare
	redshift:DeleteAuthenticationProfile
	redshift:DeleteCluster
	redshift:DeleteClusterParameterGroup
	redshift:DeleteClusterSecurityGroup
	redshift:DeleteClusterSnapshot
	redshift:DeleteClusterSubnetGroup
	redshift:DeleteCustomDomainAssociation
	redshift:DeleteEndpointAccess
	redshift:DeleteEventSubscription
	redshift:DeleteHsmClientCertificate
	redshift:DeleteHsmConfiguration
	redshift:DeletePartner
	redshift:DeleteRedshiftIdcApplication

服务前缀	操作
	redshift:DeleteResourcePolicy
	redshift:DeleteScheduledAction
	redshift:DeleteSnapshotCopyGrant
	redshift:DeleteSnapshotSchedule
	redshift:DeleteUsageLimit
	redshift:DeregisterNamespace
	redshift:DescribeAccountAttributes
	redshift:DescribeAuthenticationProfiles
	redshift:DescribeClusterDbRevisions
	redshift:DescribeClusterParameterGroups
	redshift:DescribeClusterParameters
	redshift:DescribeClusterSecurityGroups
	redshift:DescribeClusterSnapshots
	redshift:DescribeClusterSubnetGroups
	redshift:DescribeClusterTracks
	redshift:DescribeClusterVersions
	redshift:DescribeClusters
	redshift:DescribeCustomDomainAssociations
	redshift:DescribeDataShares
	redshift:DescribeDataSharesForConsumer
	redshift:DescribeDataSharesForProducer

服务前缀	操作
	redshift:DescribeDefaultClusterParameters
	redshift:DescribeEndpointAccess
	redshift:DescribeEndpointAuthorization
	redshift:DescribeEventCategories
	redshift:DescribeEventSubscriptions
	redshift:DescribeEvents
	redshift:DescribeHsmClientCertificates
	redshift:DescribeHsmConfigurations
	redshift:DescribeInboundIntegrations
	redshift:DescribeIntegrations
	redshift:DescribeLoggingStatus
	redshift:DescribeNodeConfigurationOptions
	redshift:DescribeOrderableClusterOptions
	redshift:DescribePartners
	redshift:DescribeRedshiftIdcApplications
	redshift:DescribeReservedNodeExchangeStatus
	redshift:DescribeReservedNodeOfferings
	redshift:DescribeReservedNodes
	redshift:DescribeResize
	redshift:DescribeScheduledActions
	redshift:DescribeSnapshotCopyGrants

服务前缀	操作
	redshift:DescribeSnapshotSchedules
	redshift:DescribeStorage
	redshift:DescribeTableRestoreStatus
	redshift:DescribeUsageLimits
	redshift:DisableLogging
	redshift:DisableSnapshotCopy
	redshift:DisassociateDataShareConsumer
	redshift:EnableLogging
	redshift:EnableSnapshotCopy
	redshift:FailoverPrimaryCompute
	redshift:GetClusterCredentials
	redshift:GetClusterCredentialsWithIAM
	redshift:GetIdentityCenterAuthToken
	redshift:GetReservedNodeExchangeConfigurationOptions
	redshift:GetReservedNodeExchangeOfferings
	redshift:GetResourcePolicy
	redshift:ListRecommendations
	redshift:ModifyAquaConfiguration
	redshift:ModifyAuthenticationProfile
	redshift:ModifyCluster
	redshift:ModifyClusterDbRevision

服务前缀	操作
	redshift:ModifyClusterIamRoles
	redshift:ModifyClusterMaintenance
	redshift:ModifyClusterParameterGroup
	redshift:ModifyClusterSnapshot
	redshift:ModifyClusterSnapshotSchedule
	redshift:ModifyClusterSubnetGroup
	redshift:ModifyCustomDomainAssociation
	redshift:ModifyEndpointAccess
	redshift:ModifyEventSubscription
	redshift:ModifyRedshiftIdcApplication
	redshift:ModifyScheduledAction
	redshift:ModifySnapshotCopyRetentionPeriod
	redshift:ModifySnapshotSchedule
	redshift:ModifyUsageLimit
	redshift:PauseCluster
	redshift:PurchaseReservedNodeOffering
	redshift:PutResourcePolicy
	redshift:RebootCluster
	redshift:RegisterNamespace
	redshift:RejectDataShare
	redshift:ResetClusterParameterGroup

服务前缀	操作
	redshift:ResizeCluster redshift:RestoreFromClusterSnapshot redshift:RestoreTableFromClusterSnapshot redshift:ResumeCluster redshift:RevokeClusterSecurityGroupIngress redshift:RevokeEndpointAccess redshift:RevokeSnapshotAccess redshift:RotateEncryptionKey redshift:UpdatePartnerStatus
redshift-data	redshift-data:BatchExecuteStatement redshift-data:CancelStatement redshift-data:DescribeStatement redshift-data:DescribeTable redshift-data:ExecuteStatement redshift-data:GetStatementResult redshift-data:ListDatabases redshift-data:ListSchemas redshift-data:ListStatements redshift-data:ListTables

服务前缀	操作
refactor-spaces	refactor-spaces:CreateApplication
	refactor-spaces:CreateEnvironment
	refactor-spaces:CreateRoute
	refactor-spaces:CreateService
	refactor-spaces:DeleteApplication
	refactor-spaces:DeleteEnvironment
	refactor-spaces:DeleteResourcePolicy
	refactor-spaces:DeleteRoute
	refactor-spaces:DeleteService
	refactor-spaces:GetApplication
	refactor-spaces:GetEnvironment
	refactor-spaces:GetResourcePolicy
	refactor-spaces:GetRoute
	refactor-spaces:GetService
	refactor-spaces:ListApplications
	refactor-spaces:ListEnvironmentVpcs
	refactor-spaces:ListEnvironments
	refactor-spaces:ListRoutes
	refactor-spaces:ListServices
	refactor-spaces:PutResourcePolicy
	refactor-spaces:UpdateRoute

服务前缀	操作
rekognition	rekognition:AssociateFaces rekognition:CompareFaces rekognition:CopyProjectVersion rekognition:CreateCollection rekognition:CreateDataset rekognition:CreateFaceLivenessSession rekognition:CreateProject rekognition:CreateProjectVersion rekognition:CreateStreamProcessor rekognition:CreateUser rekognition:DeleteCollection rekognition:DeleteDataset rekognition:DeleteFaces rekognition:DeleteProject rekognition:DeleteProjectPolicy rekognition:DeleteProjectVersion rekognition:DeleteStreamProcessor rekognition:DeleteUser rekognition:DescribeCollection rekognition:DescribeDataset rekognition:DescribeProjectVersions

服务前缀	操作
	rekognition:DescribeProjects rekognition:DescribeStreamProcessor rekognition:DetectCustomLabels rekognition:DetectFaces rekognition:DetectLabels rekognition:DetectModerationLabels rekognition:DetectProtectiveEquipment rekognition:DetectText rekognition:DisassociateFaces rekognition:DistributeDatasetEntries rekognition:GetCelebrityInfo rekognition:GetCelebrityRecognition rekognition:GetContentModeration rekognition:GetFaceDetection rekognition:GetFaceLivenessSessionResults rekognition:GetFaceSearch rekognition:GetLabelDetection rekognition:GetMediaAnalysisJob rekognition:GetPersonTracking rekognition:GetSegmentDetection rekognition:GetTextDetection

服务前缀	操作
	rekognition:IndexFaces rekognition:ListCollections rekognition:ListDatasetEntries rekognition:ListDatasetLabels rekognition:ListFaces rekognition:ListMediaAnalysisJobs rekognition:ListProjectPolicies rekognition:ListStreamProcessors rekognition:ListUsers rekognition:PutProjectPolicy rekognition:RecognizeCelebrities rekognition:SearchFaces rekognition:SearchFacesByImage rekognition:SearchUsers rekognition:SearchUsersByImage rekognition:StartCelebrityRecognition rekognition:StartContentModeration rekognition:StartFaceDetection rekognition:StartFaceLivenessSession rekognition:StartFaceSearch rekognition:StartLabelDetection

服务前缀	操作
	rekognition:StartMediaAnalysisJob rekognition:StartPersonTracking rekognition:StartProjectVersion rekognition:StartSegmentDetection rekognition:StartStreamProcessor rekognition:StartTextDetection rekognition:StopProjectVersion rekognition:StopStreamProcessor rekognition:UpdateDatasetEntries rekognition:UpdateStreamProcessor

服务前缀	操作
resiliencehub	resiliencehub:AcceptResourceGroupingRecommendations
	resiliencehub:AddDraftAppVersionResourceMappings
	resiliencehub:BatchUpdateRecommendationStatus
	resiliencehub:CreateApp
	resiliencehub:CreateAppVersionAppComponent
	resiliencehub:CreateAppVersionResource
	resiliencehub:CreateRecommendationTemplate
	resiliencehub:CreateResiliencyPolicy
	resiliencehub:DeleteApp
	resiliencehub:DeleteAppAssessment
	resiliencehub:DeleteAppInputSource
	resiliencehub:DeleteAppVersionAppComponent
	resiliencehub:DeleteAppVersionResource
	resiliencehub:DeleteRecommendationTemplate
	resiliencehub:DeleteResiliencyPolicy
	resiliencehub:DescribeApp
	resiliencehub:DescribeAppAssessment
	resiliencehub:DescribeAppVersion
	resiliencehub:DescribeAppVersionAppComponent
	resiliencehub:DescribeAppVersionResource
	resiliencehub:DescribeAppVersionResourcesResolutionStatus

服务前缀	操作
	resiliencehub:DescribeAppVersionTemplate
	resiliencehub:DescribeDraftAppVersionResourcesImportStatus
	resiliencehub:DescribeMetricsExport
	resiliencehub:DescribeResiliencyPolicy
	resiliencehub:DescribeResourceGroupingRecommendationTask
	resiliencehub:ImportResourcesToDraftAppVersion
	resiliencehub:ListAlarmRecommendations
	resiliencehub:ListAppAssessmentComplianceDrifts
	resiliencehub:ListAppAssessmentResourceDrifts
	resiliencehub:ListAppAssessments
	resiliencehub:ListAppComponentCompliances
	resiliencehub:ListAppComponentRecommendations
	resiliencehub:ListAppInputSources
	resiliencehub:ListAppVersionAppComponents
	resiliencehub:ListAppVersionResourceMappings
	resiliencehub:ListAppVersionResources
	resiliencehub:ListAppVersions
	resiliencehub:ListApps
	resiliencehub:ListMetrics
	resiliencehub:ListRecommendationTemplates
	resiliencehub:ListResiliencyPolicies

服务前缀	操作
	resiliencehub:ListResourceGroupingRecommendations
	resiliencehub:ListSopRecommendations
	resiliencehub:ListSuggestedResiliencyPolicies
	resiliencehub:ListTestRecommendations
	resiliencehub:ListUnsupportedAppVersionResources
	resiliencehub:PublishAppVersion
	resiliencehub:PutDraftAppVersionTemplate
	resiliencehub:RejectResourceGroupingRecommendations
	resiliencehub:RemoveDraftAppVersionResourceMappings
	resiliencehub:ResolveAppVersionResources
	resiliencehub:StartAppAssessment
	resiliencehub:StartResourceGroupingRecommendationTask
	resiliencehub:UpdateApp
	resiliencehub:UpdateAppVersion
	resiliencehub:UpdateAppVersionAppComponent
	resiliencehub:UpdateAppVersionResource
	resiliencehub:UpdateResiliencyPolicy

服务前缀	操作
resource-explorer-2	resource-explorer-2:AssociateDefaultView resource-explorer-2:BatchGetView resource-explorer-2:CreateIndex resource-explorer-2:CreateResourceExplorerSetup resource-explorer-2:CreateView resource-explorer-2:DeleteIndex resource-explorer-2:DeleteResourceExplorerSetup resource-explorer-2:DeleteView resource-explorer-2:DisassociateDefaultView resource-explorer-2:GetAccountLevelServiceConfiguration resource-explorer-2:GetDefaultView resource-explorer-2:GetIndex resource-explorer-2:GetManagedView resource-explorer-2:GetResourceExplorerSetup resource-explorer-2:GetServiceIndex resource-explorer-2:GetServiceView resource-explorer-2:ListIndexes resource-explorer-2:ListIndexesForMembers resource-explorer-2:ListManagedViews resource-explorer-2:ListServiceIndexes resource-explorer-2:ListServiceViews

服务前缀	操作
	resource-explorer-2:ListStreamingAccessForServices resource-explorer-2:ListSupportedResourceTypes resource-explorer-2:ListViews resource-explorer-2:Search resource-explorer-2:UpdateIndexType resource-explorer-2:UpdateView

服务前缀	操作
resource-groups	resource-groups:CancelTagSyncTask
	resource-groups:GetAccountSettings
	resource-groups:GetGroup
	resource-groups:GetGroupConfiguration
	resource-groups:GetGroupQuery
	resource-groups:GetTagSyncTask
	resource-groups:GroupResources
	resource-groups:ListGroupResources
	resource-groups:ListGroupingStatuses
	resource-groups:ListGroups
	resource-groups:ListTagSyncTasks
	resource-groups:PutGroupConfiguration
	resource-groups:SearchResources
	resource-groups:StartTagSyncTask
	resource-groups:UngroupResources
	resource-groups:UpdateAccountSettings
	resource-groups:UpdateGroup
	resource-groups:UpdateGroupQuery

服务前缀	操作
robomaker	robomaker:BatchDeleteWorlds robomaker:BatchDescribeSimulationJob robomaker:CancelDeploymentJob robomaker:CancelSimulationJob robomaker:CancelSimulationJobBatch robomaker:CancelWorldExportJob robomaker:CancelWorldGenerationJob robomaker:CreateDeploymentJob robomaker:CreateFleet robomaker:CreateRobot robomaker:CreateRobotApplication robomaker:CreateRobotApplicationVersion robomaker:CreateSimulationApplication robomaker:CreateSimulationApplicationVersion robomaker:CreateSimulationJob robomaker:CreateWorldExportJob robomaker:CreateWorldGenerationJob robomaker:CreateWorldTemplate RoboMaker: DeleteFleet robomaker:DeleteRobot robomaker:DeleteRobotApplication

服务前缀	操作
	robomaker:DeleteSimulationApplication robomaker:DeleteWorldTemplate robomaker:DeregisterRobot robomaker:DescribeDeploymentJob robomaker:DescribeFleet robomaker:DescribeRobot robomaker:DescribeRobotApplication robomaker:DescribeSimulationApplication robomaker:DescribeSimulationJob robomaker:DescribeSimulationJobBatch robomaker:DescribeWorld robomaker:DescribeWorldExportJob robomaker:DescribeWorldGenerationJob robomaker:DescribeWorldTemplate robomaker:GetWorldTemplateBody robomaker:ListDeploymentJobs robomaker:ListFleets robomaker:ListRobotApplications robomaker:ListRobots robomaker:ListSimulationApplications robomaker:ListSimulationJobBatches

服务前缀	操作
	robomaker:ListSimulationJobs robomaker:ListWorldExportJobs robomaker:ListWorldGenerationJobs robomaker:ListWorldTemplates robomaker:ListWorlds robomaker:RegisterRobot robomaker:RestartSimulationJob robomaker:StartSimulationJobBatch robomaker:SyncDeploymentJob robomaker:UpdateRobotApplication robomaker:UpdateSimulationApplication robomaker:UpdateWorldTemplate

服务前缀	操作
rolesanywhere	rolesanywhere:CreateProfile
	rolesanywhere:CreateTrustAnchor
	rolesanywhere:DeleteAttributeMapping
	rolesanywhere:DeleteCrl
	rolesanywhere:DeleteProfile
	rolesanywhere:DeleteTrustAnchor
	rolesanywhere:DisableCrl
	rolesanywhere:DisableProfile
	rolesanywhere:DisableTrustAnchor
	rolesanywhere:EnableCrl
	rolesanywhere:EnableProfile
	rolesanywhere:EnableTrustAnchor
	rolesanywhere:GetCrl
	rolesanywhere:GetProfile
	rolesanywhere:GetSubject
	rolesanywhere:GetTrustAnchor
	rolesanywhere:ImportCrl
	rolesanywhere:ListCrls
	rolesanywhere:ListProfiles
	rolesanywhere:ListSubjects
	rolesanywhere:ListTrustAnchors

服务前缀	操作
	rolesanywhere:PutAttributeMapping
	rolesanywhere:PutNotificationSettings
	rolesanywhere:ResetNotificationSettings
	rolesanywhere:UpdateCrl
	rolesanywhere:UpdateProfile
	rolesanywhere:UpdateTrustAnchor

服务前缀	操作
route53	route53:ActivateKeySigningKey route53:AssociateVPCWithHostedZone route53:ChangeCidrCollection route53:ChangeResourceRecordSets route53:CreateCidrCollection route53:CreateHealthCheck route53:CreateHostedZone route53:CreateKeySigningKey route53:CreateQueryLoggingConfig route53:CreateReusableDelegationSet route53:CreateTrafficPolicy route53:CreateTrafficPolicyInstance route53:CreateTrafficPolicyVersion route53:CreateVPCAssociationAuthorization route53:DeactivateKeySigningKey route53:DeleteCidrCollection route53:DeleteHealthCheck route53:DeleteHostedZone route53:DeleteKeySigningKey route53:DeleteQueryLoggingConfig route53:DeleteReusableDelegationSet

服务前缀	操作
	route53:DeleteTrafficPolicy route53:DeleteTrafficPolicyInstance route53:DeleteVPCAssociationAuthorization route53:DisableHostedZoneDNSSEC route53:DisassociateVPCFromHostedZone route53:EnableHostedZoneDNSSEC route53:GetAccountLimit route53:GetChange route53:GetCheckerIpRanges route53:GetDNSSEC route53:GetGeoLocation route53:GetHealthCheck route53:GetHealthCheckCount route53:GetHealthCheckLastFailureReason route53:GetHealthCheckStatus route53:GetHostedZone route53:GetHostedZoneCount route53:GetHostedZoneLimit route53:GetQueryLoggingConfig route53:GetReusableDelegationSet route53:GetReusableDelegationSetLimit

服务前缀	操作
	route53:GetTrafficPolicy route53:GetTrafficPolicyInstance route53:GetTrafficPolicyInstanceCount route53:ListCidrBlocks route53:ListCidrCollections route53:ListCidrLocations route53:ListGeoLocations route53:ListHealthChecks route53:ListHostedZones route53:ListHostedZonesByName route53:ListHostedZonesByVPC route53:ListQueryLoggingConfigs route53:ListResourceRecordSets route53:ListReusableDelegationSets route53:ListTrafficPolicies route53:ListTrafficPolicyInstances route53:ListTrafficPolicyInstancesByHostedZone route53:ListTrafficPolicyInstancesByPolicy route53:ListTrafficPolicyVersions route53:ListVPCAssociationAuthorizations route53:TestDNSAnswer

服务前缀	操作
	route53:UpdateHealthCheck route53:UpdateHostedZoneComment route53:UpdateTrafficPolicyComment route53:UpdateTrafficPolicyInstance

服务前缀	操作
route53-recovery-control-config	route53-recovery-control-config:CreateCluster route53-recovery-control-config:CreateControlPanel route53-recovery-control-config:CreateRoutingControl route53-recovery-control-config:CreateSafetyRule route53-recovery-control-config:DeleteCluster route53-recovery-control-config:DeleteControlPanel route53-recovery-control-config:DeleteRoutingControl route53-recovery-control-config:DeleteSafetyRule route53-recovery-control-config:DescribeCluster route53-recovery-control-config:DescribeControlPanel route53-recovery-control-config:DescribeRoutingControl route53-recovery-control-config:DescribeSafetyRule route53-recovery-control-config:GetResourcePolicy route53-recovery-control-config:ListAssociatedRoute53HealthChecks route53-recovery-control-config:ListClusters route53-recovery-control-config:ListControlPanels route53-recovery-control-config:ListRoutingControls route53-recovery-control-config:ListSafetyRules route53-recovery-control-config:UpdateCluster route53-recovery-control-config:UpdateControlPanel

服务前缀	操作
	route53-recovery-control-config:UpdateRoutingControl route53-recovery-control-config:UpdateSafetyRule

服务前缀	操作
route53-recovery-readiness	route53-recovery-readiness:CreateCell route53-recovery-readiness:CreateCrossAccountAuthorization route53-recovery-readiness:CreateReadinessCheck route53-recovery-readiness:CreateRecoveryGroup route53-recovery-readiness:CreateResourceSet route53-recovery-readiness:DeleteCell route53-recovery-readiness:DeleteCrossAccountAuthorization route53-recovery-readiness:DeleteReadinessCheck route53-recovery-readiness:DeleteRecoveryGroup route53-recovery-readiness:DeleteResourceSet route53-recovery-readiness:GetArchitectureRecommendations route53-recovery-readiness:GetCell route53-recovery-readiness:GetCellReadinessSummary route53-recovery-readiness:GetReadinessCheck route53-recovery-readiness:GetReadinessCheckResourceStatus route53-recovery-readiness:GetReadinessCheckStatus route53-recovery-readiness:GetRecoveryGroup route53-recovery-readiness:GetRecoveryGroupReadinessSummary route53-recovery-readiness:GetResourceSet route53-recovery-readiness:ListCells route53-recovery-readiness:ListCrossAccountAuthorizations

服务前缀	操作
	route53-recovery-readiness:ListReadinessChecks route53-recovery-readiness:ListRecoveryGroups route53-recovery-readiness:ListResourceSets route53-recovery-readiness:ListRules route53-recovery-readiness:UpdateCell route53-recovery-readiness:UpdateReadinessCheck route53-recovery-readiness:UpdateRecoveryGroup route53-recovery-readiness:UpdateResourceSet

服务前缀	操作
route53resolver	route53resolver:AssociateFirewallRuleGroup
	route53resolver:AssociateResolverEndpointIpAddress
	route53resolver:AssociateResolverQueryLogConfig
	route53resolver:AssociateResolverRule
	route53resolver:CreateFirewallDomainList
	route53resolver:CreateFirewallRule
	route53resolver:CreateFirewallRuleGroup
	route53resolver:CreateResolverEndpoint
	route53resolver:CreateResolverQueryLogConfig
	route53resolver:CreateResolverRule
	route53resolver:DeleteFirewallDomainList
	route53resolver:DeleteFirewallRule
	route53resolver:DeleteFirewallRuleGroup
	route53resolver:DeleteOutpostResolver
	route53resolver:DeleteResolverEndpoint
	route53resolver:DeleteResolverQueryLogConfig
	route53resolver:DeleteResolverRule
	route53resolver:DisassociateFirewallRuleGroup
	route53resolver:DisassociateResolverEndpointIpAddress
	route53resolver:DisassociateResolverQueryLogConfig
	route53resolver:DisassociateResolverRule

服务前缀	操作
	route53resolver:GetFirewallConfig
	route53resolver:GetFirewallDomainList
	route53resolver:GetFirewallRuleGroup
	route53resolver:GetFirewallRuleGroupAssociation
	route53resolver:GetFirewallRuleGroupPolicy
	route53resolver:GetOutpostResolver
	route53resolver:GetResolverConfig
	route53resolver:GetResolverDnssecConfig
	route53resolver:GetResolverEndpoint
	route53resolver:GetResolverQueryLogConfig
	route53resolver:GetResolverQueryLogConfigAssociation
	route53resolver:GetResolverQueryLogConfigPolicy
	route53resolver:GetResolverRule
	route53resolver:GetResolverRuleAssociation
	route53resolver:GetResolverRulePolicy
	route53resolver:ImportFirewallDomains
	route53resolver:ListFirewallConfigs
	route53resolver:ListFirewallDomainLists
	route53resolver:ListFirewallDomains
	route53resolver:ListFirewallRuleGroupAssociations
	route53resolver:ListFirewallRuleGroups

服务前缀	操作
	route53resolver:ListFirewallRules
	route53resolver:ListOutpostResolvers
	route53resolver:ListResolverConfigs
	route53resolver:ListResolverDnssecConfigs
	route53resolver:ListResolverEndpointIpAddresses
	route53resolver:ListResolverEndpoints
	route53resolver:ListResolverQueryLogConfigAssociations
	route53resolver:ListResolverQueryLogConfigs
	route53resolver:ListResolverRuleAssociations
	route53resolver:ListResolverRules
	route53resolver:PutFirewallRuleGroupPolicy
	route53resolver:PutResolverQueryLogConfigPolicy
	route53resolver:UpdateFirewallConfig
	route53resolver:UpdateFirewallDomains
	route53resolver:UpdateFirewallRule
	route53resolver:UpdateFirewallRuleGroupAssociation
	route53resolver:UpdateOutpostResolver
	route53resolver:UpdateResolverConfig
	route53resolver:UpdateResolverDnssecConfig
	route53resolver:UpdateResolverEndpoint
	route53resolver:UpdateResolverRule

服务前缀	操作
rum	rum:BatchCreateRumMetricDefinitions rum:BatchDeleteRumMetricDefinitions rum:BatchGetRumMetricDefinitions rum:CreateAppMonitor rum:DeleteAppMonitor rum:DeleteResourcePolicy rum:DeleteRumMetricsDestination rum:GetAppMonitor rum:GetAppMonitorData rum:GetResourcePolicy rum:ListAppMonitors rum:ListRumMetricsDestinations rum:PutResourcePolicy rum:PutRumMetricsDestination rum:UpdateAppMonitor rum:UpdateRumMetricDefinition

服务前缀	操作
S3	s3:AssociateAccessGrantsIdentityCenter
	s3:CreateAccessGrant
	s3:CreateAccessGrantsInstance
	s3:CreateAccessGrantsLocation
	s3:CreateAccessPoint
	s3:CreateAccessPointForObjectLambda
	s3:CreateBucket
	s3:CreateBucketMetadataTableConfiguration
	s3:CreateJob
	s3:CreateMultiRegionAccessPoint
	s3:DeleteAccessGrant
	s3:DeleteAccessGrantsInstance
	s3:DeleteAccessGrantsInstanceResourcePolicy
	s3:DeleteAccessGrantsLocation
	s3:DeleteAccessPoint
	s3:DeleteAccessPointForObjectLambda
	s3:DeleteAccessPointPolicy
	s3:DeleteAccessPointPolicyForObjectLambda
	s3:DeleteBucket
	s3:DeleteBucketMetadataTableConfiguration
	s3:DeleteBucketPolicy

服务前缀	操作
	s3:DeleteBucketWebsite
	s3:DeleteMultiRegionAccessPoint
	s3:DeleteStorageLensConfiguration
	s3:DescribeJob
	s3:DescribeMultiRegionAccessPointOperation
	s3:DissociateAccessGrantsIdentityCenter
	s3:GetAccelerateConfiguration
	s3:GetAccessGrant
	s3:GetAccessGrantsInstance
	s3:GetAccessGrantsInstanceForPrefix
	s3:GetAccessGrantsInstanceResourcePolicy
	s3:GetAccessGrantsLocation
	s3:GetAccessPoint
	s3:GetAccessPointConfigurationForObjectLambda
	s3:GetAccessPointForObjectLambda
	s3:GetAccessPointPolicy
	s3:GetAccessPointPolicyForObjectLambda
	s3:GetAccessPointPolicyStatus
	s3:GetAccessPointPolicyStatusForObjectLambda
	s3:GetAccountPublicAccessBlock
	s3:GetAnalyticsConfiguration

服务前缀	操作
	s3:GetBucketAbac
	s3:GetBucketAcl
	s3:GetBucketCORS
	s3:GetBucketLocation
	s3:GetBucketLogging
	s3:GetBucketNotification
	s3:GetBucketObjectLockConfiguration
	s3:GetBucketOwnershipControls
	s3:GetBucketPolicy
	s3:GetBucketPolicyStatus
	s3:GetBucketPublicAccessBlock
	s3:GetBucketRequestPayment
	s3:GetBucketVersioning
	s3:GetBucketWebsite
	s3:GetDataAccess
	s3:GetEncryptionConfiguration
	s3:GetIntelligentTieringConfiguration
	s3:GetInventoryConfiguration
	s3:GetLifecycleConfiguration
	s3:GetMetricsConfiguration
	s3:GetMultiRegionAccessPoint

服务前缀	操作
	s3:GetMultiRegionAccessPointPolicy
	s3:GetMultiRegionAccessPointPolicyStatus
	s3:GetMultiRegionAccessPointRoutes
	s3:GetReplicationConfiguration
	s3:GetStorageLensConfiguration
	s3:GetStorageLensDashboard
	s3:ListAccessGrants
	s3:ListAccessGrantsInstances
	s3:ListAccessGrantsLocations
	s3:ListAccessPoints
	s3:ListAccessPointsForObjectLambda
	s3:ListAllMyBuckets
	s3:ListBucketMultipartUploads
	s3:ListCallerAccessGrants
	s3:ListJobs
	s3:ListMultiRegionAccessPoints
	s3:ListStorageLensConfigurations
	s3:PutAccelerateConfiguration
	s3:PutAccessGrantsInstanceResourcePolicy
	s3:PutAccessPointConfigurationForObjectLambda
	s3:PutAccessPointPolicy

服务前缀	操作
	s3:PutAccessPointPolicyForObjectLambda
	s3:PutAccountPublicAccessBlock
	s3:PutAnalyticsConfiguration
	s3:PutBucketAbac
	s3:PutBucketAcl
	s3:PutBucketCORS
	s3:PutBucketLogging
	s3:PutBucketNotification
	s3:PutBucketObjectLockConfiguration
	s3:PutBucketOwnershipControls
	s3:PutBucketPolicy
	s3:PutBucketPublicAccessBlock
	s3:PutBucketRequestPayment
	s3:PutBucketVersioning
	s3:PutBucketWebsite
	s3:PutEncryptionConfiguration
	s3:PutIntelligentTieringConfiguration
	s3:PutInventoryConfiguration
	s3:PutLifecycleConfiguration
	s3:PutMetricsConfiguration
	s3:PutMultiRegionAccessPointPolicy

服务前缀	操作
	s3:PutReplicationConfiguration s3:PutStorageLensConfiguration s3:SubmitMultiRegionAccessPointRoutes s3:UpdateAccessGrantsLocation s3:UpdateBucketMetadataJournalTableConfiguration s3:UpdateJobPriority s3:UpdateJobStatus
s3-outposts	s3-outposts:CreateEndpoint s3-outposts:DeleteEndpoint s3-outposts:ListEndpoints s3-outposts:ListOutpostsWithS3 s3-outposts:ListSharedEndpoints

服务前缀	操作
sagemaker-geospatial	sagemaker-geospatial:DeleteEarthObservationJob
	sagemaker-geospatial:DeleteVectorEnrichmentJob
	sagemaker-geospatial:ExportEarthObservationJob
	sagemaker-geospatial:ExportVectorEnrichmentJob
	sagemaker-geospatial:GetEarthObservationJob
	sagemaker-geospatial:GetRasterDataCollection
	sagemaker-geospatial:GetTile
	sagemaker-geospatial:GetVectorEnrichmentJob
	sagemaker-geospatial:ListEarthObservationJobs
	sagemaker-geospatial:ListRasterDataCollections
	sagemaker-geospatial:ListVectorEnrichmentJobs
	sagemaker-geospatial:SearchRasterDataCollection
	sagemaker-geospatial:StartEarthObservationJob
	sagemaker-geospatial:StartVectorEnrichmentJob
	sagemaker-geospatial:StopEarthObservationJob
	sagemaker-geospatial:StopVectorEnrichmentJob

服务前缀	操作
savingsplans	savingsplans:CreateSavingsPlan savingsplans:DeleteQueuedSavingsPlan savingsplans:DescribeSavingsPlanRates savingsplans:DescribeSavingsPlans savingsplans:DescribeSavingsPlansOfferingRates savingsplans:DescribeSavingsPlansOfferings savingsplans:ReturnSavingsPlan

服务前缀	操作
schemas	schemas:CreateDiscoverer
	schemas:CreateRegistry
	schemas:CreateSchema
	schemas:DeleteDiscoverer
	schemas:DeleteRegistry
	schemas:DeleteResourcePolicy
	schemas:DeleteSchema
	schemas:DeleteSchemaVersion
	schemas:DescribeCodeBinding
	schemas:DescribeDiscoverer
	schemas:DescribeRegistry
	schemas:DescribeSchema
	schemas:ExportSchema
	schemas:GetCodeBindingSource
	schemas:GetDiscoveredSchema
	schemas:GetResourcePolicy
	schemas:ListDiscoverers
	schemas:ListRegistries
	schemas:ListSchemaVersions
	schemas:ListSchemas
	schemas:PutCodeBinding

服务前缀	操作
	schemas:PutResourcePolicy schemas:SearchSchemas schemas:StartDiscoverer schemas:StopDiscoverer schemas:UpdateDiscoverer schemas:UpdateRegistry schemas:UpdateSchema
sdb	sdb:CreateDomain sdb:DeleteDomain sdb:DomainMetadata sdb:ListDomains

服务前缀	操作
secretsmanager	secretsmanager:CancelRotateSecret secretsmanager:CreateSecret secretsmanager:DeleteResourcePolicy secretsmanager:DeleteSecret secretsmanager:DescribeSecret secretsmanager:GetRandomPassword secretsmanager:GetResourcePolicy secretsmanager:GetSecretValue secretsmanager:ListSecretVersionIds secretsmanager:ListSecrets secretsmanager:PutResourcePolicy secretsmanager:PutSecretValue secretsmanager:RemoveRegionsFromReplication secretsmanager:ReplicateSecretToRegions secretsmanager:RestoreSecret secretsmanager:RotateSecret secretsmanager:StopReplicationToReplica secretsmanager:UpdateSecret secretsmanager:ValidateResourcePolicy

服务前缀	操作
securityhub	securityhub:AcceptAdministratorInvitation securityhub:AcceptInvitation securityhub:BatchDeleteAutomationRules securityhub:BatchDisableStandards securityhub:BatchEnableStandards securityhub:BatchGetAutomationRules securityhub:BatchGetConfigurationPolicyAssociations securityhub:BatchGetSecurityControls securityhub:BatchGetStandardsControlAssociations securityhub:BatchImportFindings securityhub:BatchUpdateAutomationRules securityhub:BatchUpdateFindings securityhub:BatchUpdateStandardsControlAssociations securityhub:ConnectorRegistrationsV2 securityhub:CreateActionTarget securityhub:CreateAggregatorV2 securityhub:CreateAutomationRule securityhub:CreateAutomationRuleV2 securityhub:CreateConfigurationPolicy securityhub:CreateConnectorV2 securityhub:CreateFindingAggregator

服务前缀	操作
	securityhub:CreateInsight securityhub:CreateMembers securityhub:CreateTicketV2 securityhub:DeclineInvitations securityhub:DeleteActionTarget securityhub:DeleteAggregatorV2 securityhub:DeleteAutomationRuleV2 securityhub:DeleteConfigurationPolicy securityhub:DeleteConnectorV2 securityhub:DeleteFindingAggregator securityhub:DeleteInsight securityhub:DeleteInvitations securityhub:DeleteMembers securityhub:DescribeActionTargets securityhub:DescribeHub securityhub:DescribeOrganizationConfiguration securityhub:DescribeProducts securityhub:DescribeSecurityHubV2 securityhub:DescribeStandards securityhub:DisableImportFindingsForProduct securityhub:DisableOrganizationAdminAccount

服务前缀	操作
	securityhub:DisableSecurityHub securityhub:DisableSecurityHubV2 securityhub:DisassociateFromAdministratorAccount securityhub:DisassociateFromMasterAccount securityhub:DisassociateMembers securityhub:EnableImportFindingsForProduct securityhub:EnableOrganizationAdminAccount securityhub:EnableSecurityHub securityhub:GetAdministratorAccount securityhub:GetAggregatorV2 securityhub:GetAutomationRuleV2 securityhub:GetConfigurationPolicy securityhub:GetConfigurationPolicyAssociation securityhub:GetConnectorV2 securityhub:GetEnabledStandards securityhub:GetFindingAggregator securityhub:GetFindingHistory securityhub:GetFindings securityhub:GetInsightResults securityhub:GetInsights securityhub:GetInvitationsCount

服务前缀	操作
	securityhub:GetMasterAccount securityhub:GetMembers securityhub:GetSecurityControlDefinition securityhub:InviteMembers securityhub:ListAggregatorsV2 securityhub:ListAutomationRules securityhub:ListAutomationRulesV2 securityhub:ListConfigurationPolicies securityhub:ListConfigurationPolicyAssociations securityhub:ListConnectorsV2 securityhub:ListEnabledProductsForImport securityhub:ListFindingAggregators securityhub:ListInvitations securityhub:ListMembers securityhub:ListOrganizationAdminAccounts securityhub:ListSecurityControlDefinitions securityhub:ListStandardsControlAssociations securityhub:StartConfigurationPolicyAssociation securityhub:StartConfigurationPolicyDisassociation securityhub:UpdateActionTarget securityhub:UpdateAggregatorV2

服务前缀	操作
	securityhub:UpdateAutomationRuleV2 securityhub:UpdateConfigurationPolicy securityhub:UpdateConnectorV2 securityhub:UpdateFindingAggregator securityhub:UpdateFindings securityhub:UpdateInsight securityhub:UpdateOrganizationConfiguration securityhub:UpdateSecurityControl securityhub:UpdateSecurityHubConfiguration

服务前缀	操作
securitylake	securitylake:CreateAwsLogSource
	securitylake:CreateCustomLogSource
	securitylake:CreateDataLakeExceptionSubscription
	securitylake:CreateDataLakeOrganizationConfiguration
	securitylake:CreateSubscriber
	securitylake:CreateSubscriberNotification
	securitylake:DeleteAwsLogSource
	securitylake:DeleteCustomLogSource
	securitylake:DeleteDataLakeExceptionSubscription
	securitylake:DeleteDataLakeOrganizationConfiguration
	securitylake:DeleteSubscriber
	securitylake:DeleteSubscriberNotification
	securitylake:DeregisterDataLakeDelegatedAdministrator
	securitylake:GetDataLakeExceptionSubscription
	securitylake:GetDataLakeOrganizationConfiguration
	securitylake:GetDataLakeSources
	securitylake:GetSubscriber
	securitylake:ListDataLakes
	securitylake:ListLogSources
	securitylake:ListSubscribers
	securitylake:RegisterDataLakeDelegatedAdministrator

服务前缀	操作
	securitylake:UpdateDataLakeExceptionSubscription securitylake:UpdateSubscriber securitylake:UpdateSubscriberNotification
serverlessrepo	serverlessrepo:CreateApplication serverlessrepo:CreateApplicationVersion serverlessrepo:CreateCloudFormationChangeSet serverlessrepo:CreateCloudFormationTemplate serverlessrepo:DeleteApplication serverlessrepo:GetApplication serverlessrepo:GetApplicationPolicy serverlessrepo:GetCloudFormationTemplate serverlessrepo:ListApplicationDependencies serverlessrepo:ListApplicationVersions serverlessrepo:ListApplications serverlessrepo:PutApplicationPolicy serverlessrepo:UnshareApplication serverlessrepo:UpdateApplication

服务前缀	操作
servicecatalog	servicecatalog:AcceptPortfolioShare
	servicecatalog:AssociateBudgetWithResource
	servicecatalog:AssociatePrincipalWithPortfolio
	servicecatalog:AssociateProductWithPortfolio
	servicecatalog:AssociateServiceActionWithProvisioningArtifact
	servicecatalog:BatchAssociateServiceActionWithProvisioningArtifact
	servicecatalog:BatchDisassociateServiceActionFromProvisioningArtifact
	servicecatalog:CopyProduct
	servicecatalog:CreateAttributeGroup
	servicecatalog:CreateConstraint
	servicecatalog:CreatePortfolio
	servicecatalog:CreatePortfolioShare
	servicecatalog:CreateProduct
	servicecatalog:CreateProvisionedProductPlan
	servicecatalog:CreateProvisioningArtifact
	servicecatalog:CreateServiceAction
	servicecatalog:DeleteAttributeGroup
	servicecatalog:DeleteConstraint
	servicecatalog:DeletePortfolio
	servicecatalog:DeletePortfolioShare

服务前缀	操作
	servicecatalog:DeleteProduct
	servicecatalog:DeleteProvisionedProductPlan
	servicecatalog:DeleteProvisioningArtifact
	servicecatalog:DeleteServiceAction
	servicecatalog:DescribeConstraint
	servicecatalog:DescribeCopyProductStatus
	servicecatalog:DescribePortfolio
	servicecatalog:DescribePortfolioShareStatus
	servicecatalog:DescribePortfolioShares
	servicecatalog:DescribeProduct
	servicecatalog:DescribeProductAsAdmin
	servicecatalog:DescribeProductView
	servicecatalog:DescribeProvisionedProduct
	servicecatalog:DescribeProvisionedProductPlan
	servicecatalog:DescribeProvisioningArtifact
	servicecatalog:DescribeProvisioningParameters
	servicecatalog:DescribeRecord
	servicecatalog:DescribeServiceAction
	servicecatalog:DescribeServiceActionExecutionParameters
	servicecatalog:DisableAWSOrganizationsAccess
	servicecatalog:DisassociateBudgetFromResource

服务前缀	操作
	servicecatalog:DisassociatePrincipalFromPortfolio
	servicecatalog:DisassociateProductFromPortfolio
	servicecatalog:DisassociateServiceActionFromProvisioningArtifact
	servicecatalog:EnableAWSOrganizationsAccess
	servicecatalog:ExecuteProvisionedProductPlan
	servicecatalog:ExecuteProvisionedProductServiceAction
	servicecatalog:GetAWSOrganizationsAccessStatus
	servicecatalog:GetProvisionedProductOutputs
	servicecatalog:ImportAsProvisionedProduct
	servicecatalog:ListAcceptedPortfolioShares
	servicecatalog:ListAttributeGroups
	servicecatalog:ListBudgetsForResource
	servicecatalog:ListConstraintsForPortfolio
	servicecatalog:ListLaunchPaths
	servicecatalog:ListOrganizationPortfolioAccess
	servicecatalog:ListPortfolioAccess
	servicecatalog:ListPortfolios
	servicecatalog:ListPortfoliosForProduct
	servicecatalog:ListPrincipalsForPortfolio
	servicecatalog:ListProvisionedProductPlans
	servicecatalog:ListProvisioningArtifacts

服务前缀	操作
	servicecatalog:ListProvisioningArtifactsForServiceAction
	servicecatalog:ListRecordHistory
	servicecatalog:ListServiceActions
	servicecatalog:ListServiceActionsForProvisioningArtifact
	servicecatalog:ListStackInstancesForProvisionedProduct
	servicecatalog:NotifyProvisionProductEngineWorkflowResult
	servicecatalog:NotifyTerminateProvisionedProductEngine WorkflowResult
	servicecatalog:NotifyUpdateProvisionedProductEngineWor kflowResult
	servicecatalog:ProvisionProduct
	servicecatalog:RejectPortfolioShare
	servicecatalog:ScanProvisionedProducts
	servicecatalog:SearchProducts
	servicecatalog:SearchProductsAsAdmin
	servicecatalog:SearchProvisionedProducts
	servicecatalog:TerminateProvisionedProduct
	servicecatalog:UpdateConstraint
	servicecatalog:UpdatePortfolio
	servicecatalog:UpdatePortfolioShare
	servicecatalog:UpdateProduct
	servicecatalog:UpdateProvisionedProduct

服务前缀	操作
	servicecatalog:UpdateProvisionedProductProperties servicecatalog:UpdateProvisioningArtifact servicecatalog:UpdateServiceAction

服务前缀	操作
servicediscovery	servicediscovery:CreateHttpNamespace servicediscovery:CreatePrivateDnsNamespace servicediscovery:CreatePublicDnsNamespace servicediscovery:CreateService servicediscovery:DeleteNamespace servicediscovery:DeleteService servicediscovery:DeleteServiceAttributes servicediscovery:DeregisterInstance servicediscovery:GetInstance servicediscovery:GetInstancesHealthStatus servicediscovery:GetNamespace servicediscovery:GetOperation servicediscovery:GetService servicediscovery:ListInstances servicediscovery:ListNamespaces servicediscovery:ListOperations servicediscovery:ListServices servicediscovery:RegisterInstance servicediscovery:UpdateHttpNamespace servicediscovery:UpdateInstanceCustomHealthStatus servicediscovery:UpdatePrivateDnsNamespace

服务前缀	操作
	servicediscovery:UpdatePublicDnsNamespace servicediscovery:UpdateService servicediscovery:UpdateServiceAttributes

服务前缀	操作
servicequotas	servicequotas:AssociateServiceQuotaTemplate servicequotas:CreateSupportCase servicequotas:DeleteServiceQuotaIncreaseRequestFromTemplate servicequotas:DisassociateServiceQuotaTemplate servicequotas:GetAWSDefaultServiceQuota servicequotas:GetAssociationForServiceQuotaTemplate servicequotas:GetAutoManagementConfiguration servicequotas:GetQuotaUtilizationReport servicequotas:GetRequestedServiceQuotaChange servicequotas:GetServiceQuota servicequotas:GetServiceQuotaIncreaseRequestFromTemplate servicequotas:ListAWSDefaultServiceQuotas servicequotas:ListRequestedServiceQuotaChangeHistory servicequotas:ListRequestedServiceQuotaChangeHistoryByQuota servicequotas:ListServiceQuotaIncreaseRequestsInTemplate servicequotas:ListServiceQuotas servicequotas:ListServices servicequotas:PutServiceQuotaIncreaseRequestIntoTemplate servicequotas:RequestServiceQuotaIncrease servicequotas:StartAutoManagement servicequotas:StartQuotaUtilizationReport

服务前缀	操作
	servicequotas:StopAutoManagement servicequotas:UpdateAutoManagement

服务前缀	操作
ses	ses:BatchGetMetricData ses:CloneReceiptRuleSet ses:CreateAddonInstance ses:CreateAddonSubscription ses:CreateAddressList ses:CreateAddressListImportJob ses:CreateArchive ses:CreateConfigurationSet ses:CreateConfigurationSetEventDestination ses:CreateConfigurationSetTrackingOptions ses:CreateContact ses:CreateContactList ses:CreateCustomVerificationEmailTemplate ses:CreateDedicatedIpPool ses:CreateDeliverabilityTestReport ses:CreateEmailIdentity ses:CreateEmailIdentityPolicy ses:CreateEmailTemplate ses:CreateImportJob ses:CreateIngressPoint ses:CreateMultiRegionEndpoint

服务前缀	操作
	ses:CreateReceiptFilter ses:CreateReceiptRule ses:CreateReceiptRuleSet ses:CreateRelay ses:CreateRuleSet ses:CreateTemplate ses:CreateTenant ses:CreateTenantResourceAssociation ses:CreateTrafficPolicy ses:DeleteAddonInstance ses:DeleteAddonSubscription ses:DeleteAddressList ses:DeleteArchive ses:DeleteConfigurationSet ses:DeleteConfigurationSetEventDestination ses:DeleteConfigurationSetTrackingOptions ses:DeleteContact ses:DeleteContactList ses:DeleteCustomVerificationEmailTemplate ses:DeleteDedicatedIpPool ses:DeleteEmailIdentity

服务前缀	操作
	ses:DeleteEmailIdentityPolicy ses:DeleteEmailTemplate ses:DeleteIdentity ses:DeleteIdentityPolicy ses:DeleteIngressPoint ses:DeleteMultiRegionEndpoint ses:DeleteReceiptFilter ses:DeleteReceiptRule ses:DeleteReceiptRuleSet ses:DeleteRelay ses:DeleteRuleSet ses:DeleteSuppressedDestination ses:DeleteTemplate ses:DeleteTenant ses:DeleteTenantResourceAssociation ses:DeleteTrafficPolicy ses:DeleteVerifiedEmailAddress ses:DeregisterMemberFromAddressList ses:DescribeActiveReceiptRuleSet ses:DescribeConfigurationSet ses:DescribeReceiptRule

服务前缀	操作
	ses:DescribeReceiptRuleSet ses:GetAccount ses:GetAccountSendingEnabled ses:GetAddonInstance ses:GetAddonSubscription ses:GetAddressList ses:GetArchive ses:GetArchiveExport ses:GetArchiveMessage ses:GetArchiveMessageContent ses:GetArchiveSearch ses:GetArchiveSearchResults ses:GetBlacklistReports ses:GetConfigurationSet ses:GetConfigurationSetEventDestinations ses:GetContact ses:GetContactList ses:GetCustomVerificationEmailTemplate ses:GetDedicatedIp ses:GetDedicatedIpPool ses:GetDedicatedIps

服务前缀	操作
	ses:GetDeliverabilityDashboardOptions ses:GetDeliverabilityTestReport ses:GetDomainDeliverabilityCampaign ses:GetDomainStatisticsReport ses:GetEmailAddressInsights ses:GetEmailIdentity ses:GetEmailIdentityPolicies ses:GetEmailTemplate ses:GetIdentityDkimAttributes ses:GetIdentityMailFromDomainAttributes ses:GetIdentityNotificationAttributes ses:GetIdentityPolicies ses:GetIdentityVerificationAttributes ses:GetImportJob ses:GetIngressPoint ses:GetMemberOfAddressList ses:GetMessageInsights ses:GetMultiRegionEndpoint ses:GetRelay ses:GetRuleSet ses:GetSendQuota

服务前缀	操作
	ses:GetSendStatistics ses:GetSuppressedDestination ses:GetTemplate ses:GetTenant ses:GetTrafficPolicy ses:ListAddonInstances ses:ListAddonSubscriptions ses:ListAddressListImportJobs ses:ListAddressLists ses:ListArchiveExports ses:ListArchiveSearches ses:ListArchives ses:ListConfigurationSets ses:ListContactLists ses:ListContacts ses:ListCustomVerificationEmailTemplates ses:ListDedicatedIpPools ses:ListDeliverabilityTestReports ses:ListDomainDeliverabilityCampaigns ses:ListEmailIdentities ses:ListEmailTemplates

服务前缀	操作
	ses:ListExportJobs ses:ListIdentities ses:ListIdentityPolicies ses:ListImportJobs ses:ListIngressPoints ses:ListMembersOfAddressList ses:ListMultiRegionEndpoints ses:ListReceiptFilters ses:ListReceiptRuleSets ses:ListRecommendations ses:ListRelays ses:ListReputationEntities ses:ListResourceTenants ses:ListRuleSets ses:ListSuppressedDestinations ses:ListTemplates ses:ListTenantResources ses:ListTenants ses:ListTrafficPolicies ses:ListVerifiedEmailAddresses ses:PutAccountDedicatedIpWarmupAttributes

服务前缀	操作
	ses:PutAccountDetails ses:PutAccountSendingAttributes ses:PutAccountSuppressionAttributes ses:PutAccountVdmAttributes ses:PutConfigurationSetArchivingOptions ses:PutConfigurationSetDeliveryOptions ses:PutConfigurationSetReputationOptions ses:PutConfigurationSetSendingOptions ses:PutConfigurationSetSuppressionOptions ses:PutConfigurationSetTrackingOptions ses:PutConfigurationSetVdmOptions ses:PutDedicatedIpInPool ses:PutDedicatedIpPoolScalingAttributes ses:PutDedicatedIpWarmupAttributes ses:PutDeliverabilityDashboardOption ses:PutEmailIdentityConfigurationSetAttributes ses:PutEmailIdentityDkimAttributes ses:PutEmailIdentityDkimSigningAttributes ses:PutEmailIdentityFeedbackAttributes ses:PutEmailIdentityMailFromAttributes ses:PutIdentityPolicy

服务前缀	操作
	ses:PutSuppressedDestination ses:RegisterMemberToAddressList ses:ReorderReceiptRuleSet ses:SendBounce ses:SendCustomVerificationEmail ses:SetActiveReceiptRuleSet ses:SetIdentityDkimEnabled ses:SetIdentityFeedbackForwardingEnabled ses:SetIdentityHeadersInNotificationsEnabled ses:SetIdentityMailFromDomain ses:SetIdentityNotificationTopic ses:SetReceiptRulePosition ses:StartArchiveExport ses:StartArchiveSearch ses:StopArchiveExport ses:StopArchiveSearch ses:TestRenderEmailTemplate ses:TestRenderTemplate ses:UpdateAccountSendingEnabled ses:UpdateArchive ses:UpdateConfigurationSetEventDestination

服务前缀	操作
	ses:UpdateConfigurationSetReputationMetricsEnabled ses:UpdateConfigurationSetSendingEnabled ses:UpdateConfigurationSetTrackingOptions ses:UpdateContact ses:UpdateContactList ses:UpdateCustomVerificationEmailTemplate ses:UpdateEmailIdentityPolicy ses:UpdateEmailTemplate ses:UpdateIngressPoint ses:UpdateReceiptRule ses:UpdateRelay ses:UpdateRuleSet ses:UpdateTemplate ses:UpdateTrafficPolicy ses:VerifyDomainDkim ses:VerifyDomainIdentity ses:VerifyEmailAddress ses:VerifyEmailIdentity

服务前缀	操作
shield	shield:AssociateDRTLogBucket
	shield:AssociateHealthCheck
	shield:AssociateProactiveEngagementDetails
	shield:CreateProtection
	shield:CreateProtectionGroup
	shield:CreateSubscription
	shield:DeleteProtection
	shield:DeleteProtectionGroup
	shield:DeleteSubscription
	shield:DescribeAttack
	shield:DescribeAttackStatistics
	shield:DescribeDRTAccess
	shield:DescribeEmergencyContactSettings
	shield:DescribeProtection
	shield:DescribeProtectionGroup
	shield:DescribeSubscription
	shield:DisableApplicationLayerAutomaticResponse
	shield:DisableProactiveEngagement
	shield:DisassociateDRTLogBucket
	shield:DisassociateDRTRole
	shield:DisassociateHealthCheck

服务前缀	操作
	shield:EnableApplicationLayerAutomaticResponse
	shield:EnableProactiveEngagement
	shield:GetSubscriptionState
	shield:ListAttacks
	shield:ListProtectionGroups
	shield:ListProtections
	shield:ListResourcesInProtectionGroup
	shield:UpdateApplicationLayerAutomaticResponse
	shield:UpdateEmergencyContactSettings
	shield:UpdateProtectionGroup
	shield:UpdateSubscription

服务前缀	操作
signer	signer:AddProfilePermission signer:CancelSigningProfile signer:DescribeSigningJob signer:GetSigningPlatform signer:GetSigningProfile signer:ListProfilePermissions signer:ListSigningJobs signer:ListSigningPlatforms signer:ListSigningProfiles signer:PutSigningProfile signer:RemoveProfilePermission signer:RevokeSignature signer:RevokeSigningProfile signer:SignPayload signer:StartSigningJob

服务前缀	操作
simspaceweaver	simspaceweaver:CreateSnapshot
	simspaceweaver:DeleteApp
	simspaceweaver:DeleteSimulation
	simspaceweaver:DescribeApp
	simspaceweaver:DescribeSimulation
	simspaceweaver:ListApps
	simspaceweaver:ListSimulations
	simspaceweaver:StartApp
	simspaceweaver:StartClock
	simspaceweaver:StartSimulation
	simspaceweaver:StopApp
	simspaceweaver:StopClock
	simspaceweaver:StopSimulation

服务前缀	操作
sms	sms:CreateApp sms:CreateReplicationJob sms:DeleteApp sms:DeleteAppLaunchConfiguration sms:DeleteAppReplicationConfiguration sms:DeleteAppValidationConfiguration sms:DeleteReplicationJob sms:DeleteServerCatalog sms:DisassociateConnector sms:GenerateChangeSet sms:GenerateTemplate sms:GetApp sms:GetAppLaunchConfiguration sms:GetAppReplicationConfiguration sms:GetAppValidationConfiguration sms:GetAppValidationOutput sms:GetConnectors sms:GetReplicationJobs sms:GetReplicationRuns sms:GetServers sms:ImportAppCatalog

服务前缀	操作
	sms:ImportServerCatalog sms:LaunchApp sms:ListApps sms:NotifyAppValidationOutput sms:PutAppLaunchConfiguration sms:PutAppReplicationConfiguration sms:PutAppValidationConfiguration sms:StartAppReplication sms:StartOnDemandAppReplication sms:StartOnDemandReplicationRun sms:StopAppReplication sms:TerminateApp sms:UpdateApp sms:UpdateReplicationJob

服务前缀	操作
sms-voice	sms-voice:AssociateProtectConfiguration
	sms-voice:CreateConfigurationSet
	sms-voice:CreateConfigurationSetEventDestination
	sms-voice:CreateEventDestination
	sms-voice:CreateOptOutList
	sms-voice:CreatePool
	sms-voice:CreateProtectConfiguration
	sms-voice:CreateRegistration
	sms-voice:CreateRegistrationAssociation
	sms-voice:CreateRegistrationAttachment
	sms-voice:CreateRegistrationVersion
	sms-voice:CreateVerifiedDestinationNumber
	sms-voice:DeleteAccountDefaultProtectConfiguration
	sms-voice:DeleteConfigurationSet
	sms-voice:DeleteConfigurationSetEventDestination
	sms-voice:DeleteDefaultMessageType
	sms-voice:DeleteDefaultSenderId
	sms-voice:DeleteEventDestination
	sms-voice:DeleteKeyword
	sms-voice:DeleteMediaMessageSpendLimitOverride
	sms-voice:DeleteOptOutList

服务前缀	操作
	sms-voice:DeleteOptedOutNumber
	sms-voice:DeletePool
	sms-voice:DeleteProtectConfiguration
	sms-voice:DeleteProtectConfigurationRuleSetNumberOverride
	sms-voice:DeleteRegistration
	sms-voice:DeleteRegistrationAttachment
	sms-voice:DeleteResourcePolicy
	sms-voice:DeleteTextMessageSpendLimitOverride
	sms-voice:DeleteVerifiedDestinationNumber
	sms-voice:DeleteVoiceMessageSpendLimitOverride
	sms-voice:DescribeAccountAttributes
	sms-voice:DescribeAccountLimits
	sms-voice:DescribeConfigurationSets
	sms-voice:DescribeKeywords
	sms-voice:DescribeOptOutLists
	sms-voice:DescribeOptedOutNumbers
	sms-voice:DescribePhoneNumbers
	sms-voice:DescribePools
	sms-voice:DescribeProtectConfigurations
	sms-voice:DescribeRegistrationAttachments
	sms-voice:DescribeRegistrationFieldDefinitions

服务前缀	操作
	sms-voice:DescribeRegistrationFieldValues
	sms-voice:DescribeRegistrationSectionDefinitions
	sms-voice:DescribeRegistrationTypeDefinitions
	sms-voice:DescribeRegistrationVersions
	sms-voice:DescribeRegistrations
	sms-voice:DescribeSenderIds
	sms-voice:DescribeSpendLimits
	sms-voice:DescribeVerifiedDestinationNumbers
	sms-voice:DisassociateOriginationIdentity
	sms-voice:DisassociateProtectConfiguration
	sms-voice:DiscardRegistrationVersion
	sms-voice:GetConfigurationSetEventDestinations
	sms-voice:GetProtectConfigurationCountryRuleSet
	sms-voice:GetResourcePolicy
	sms-voice:ListConfigurationSets
	sms-voice:ListPoolOriginationIdentities
	sms-voice:ListProtectConfigurationRuleSetNumberOverrides
	sms-voice:ListRegistrationAssociations
	sms-voice:PutKeyword
	sms-voice:PutOptedOutNumber
	sms-voice:PutProtectConfigurationRuleSetNumberOverride

服务前缀	操作
	sms-voice:PutResourcePolicy
	sms-voice:ReleasePhoneNumber
	sms-voice:ReleaseSenderId
	sms-voice:RequestPhoneNumber
	sms-voice:RequestSenderId
	sms-voice:SendDestinationNumberVerificationCode
	sms-voice:SetAccountDefaultProtectConfiguration
	sms-voice:SetDefaultMessageFeedbackEnabled
	sms-voice:SetDefaultMessageType
	sms-voice:SetDefaultSenderId
	sms-voice:SetMediaMessageSpendLimitOverride
	sms-voice:SetTextMessageSpendLimitOverride
	sms-voice:SetVoiceMessageSpendLimitOverride
	sms-voice:SubmitRegistrationVersion
	sms-voice:UpdateConfigurationSetEventDestination
	sms-voice:UpdateEventDestination
	sms-voice:UpdatePhoneNumber
	sms-voice:UpdatePool
	sms-voice:UpdateProtectConfiguration
	sms-voice:UpdateProtectConfigurationCountryRuleSet
	sms-voice:UpdateSenderId

服务前缀	操作
snowball	snowball:CancelCluster snowball:CancelJob snowball:CreateAddress snowball:CreateCluster snowball:CreateJob snowball:CreateLongTermPricing snowball:CreateReturnShippingLabel snowball:DescribeAddress snowball:DescribeAddresses snowball:DescribeCluster snowball:DescribeJob snowball:DescribeReturnShippingLabel snowball:GetJobManifest snowball:GetJobUnlockCode snowball:GetSnowballUsage snowball:GetSoftwareUpdates snowball:ListClusterJobs snowball:ListClusters snowball:ListCompatibleImages snowball:ListJobs snowball:ListLongTermPricing

服务前缀	操作
	snowball:ListPickupLocations snowball:ListServiceVersions snowball:UpdateCluster snowball:UpdateJob snowball:UpdateJobShipmentState snowball:UpdateLongTermPricing
sqs	sqs:AddPermission sqs:CancelMessageMoveTask sqs:CreateQueue sqs:DeleteQueue sqs:PurgeQueue sqs:RemovePermission sqs:SetQueueAttributes

服务前缀	操作
ssm	ssm:AssociateOpsItemRelatedItem ssm:CancelCommand ssm:CancelMaintenanceWindowExecution ssm:CreateActivation ssm:CreateAssociation ssm:CreateAssociationBatch ssm:CreateDocument ssm:CreateMaintenanceWindow ssm:CreateOpsItem ssm:CreateOpsMetadata ssm:CreatePatchBaseline ssm:CreateResourceDataSync ssm:DeleteActivation ssm:DeleteAssociation ssm:DeleteDocument ssm:DeleteInventory ssm:DeleteMaintenanceWindow ssm:DeleteOpsItem ssm:DeleteOpsMetadata ssm:DeleteParameter ssm:DeleteParameters

服务前缀	操作
	ssm:DeletePatchBaseline ssm:DeleteResourceDataSync ssm:DeleteResourcePolicy ssm:DeregisterManagedInstance ssm:DeregisterPatchBaselineForPatchGroup ssm:DeregisterTargetFromMaintenanceWindow ssm:DeregisterTaskFromMaintenanceWindow ssm:DescribeActivations ssm:DescribeAssociation ssm:DescribeAssociationExecutionTargets ssm:DescribeAssociationExecutions ssm:DescribeAutomationExecutions ssm:DescribeAutomationStepExecutions ssm:DescribeAvailablePatches ssm:DescribeDocument ssm:DescribeDocumentParameters ssm:DescribeDocumentPermission ssm:DescribeEffectiveInstanceAssociations ssm:DescribeEffectivePatchesForPatchBaseline ssm:DescribeInstanceAssociationsStatus ssm:DescribeInstanceInformation

服务前缀	操作
	ssm:DescribeInstancePatchStates ssm:DescribeInstancePatchStatesForPatchGroup ssm:DescribeInstancePatches ssm:DescribeInstanceProperties ssm:DescribeInventoryDeletions ssm:DescribeMaintenanceWindowExecutionTaskInvocations ssm:DescribeMaintenanceWindowExecutionTasks ssm:DescribeMaintenanceWindowExecutions ssm:DescribeMaintenanceWindowSchedule ssm:DescribeMaintenanceWindowTargets ssm:DescribeMaintenanceWindowTasks ssm:DescribeMaintenanceWindows ssm:DescribeMaintenanceWindowsForTarget ssm:DescribeOpsItems ssm:DescribeParameters ssm:DescribePatchBaselines ssm:DescribePatchGroupState ssm:DescribePatchGroups ssm:DescribePatchProperties ssm:DescribeSessions ssm:DisassociateOpsItemRelatedItem

服务前缀	操作
	ssm:GetAccessToken ssm:GetAutomationExecution ssm:GetCalendarState ssm:GetCommandInvocation ssm:GetConnectionStatus ssm:GetDefaultPatchBaseline ssm:GetDeployablePatchSnapshotForInstance ssm:GetDocument ssm:GetExecutionPreview ssm:GetInventory ssm:GetInventorySchema ssm:GetMaintenanceWindow ssm:GetMaintenanceWindowExecution ssm:GetMaintenanceWindowExecutionTask ssm:GetMaintenanceWindowExecutionTaskInvocation ssm:GetMaintenanceWindowTask ssm:GetOpsItem ssm:GetOpsMetadata ssm:GetOpsSummary ssm:GetParameter ssm:GetParameterHistory

服务前缀	操作
	ssm:GetParameters ssm:GetParametersByPath ssm:GetPatchBaseline ssm:GetPatchBaselineForPatchGroup ssm:GetResourcePolicies ssm:GetServiceSetting ssm:LabelParameterVersion ssm:ListAssociationVersions ssm:ListAssociations ssm:ListCommandInvocations ssm:ListCommands ssm:ListComplianceItems ssm:ListComplianceSummaries ssm:ListDocumentMetadataHistory ssm:ListDocumentVersions ssm:ListDocuments ssm:ListInstanceAssociations ssm:ListInventoryEntries ssm:ListNodes ssm:ListNodesSummary ssm:ListOpsItemEvents

服务前缀	操作
	ssm:ListOpsItemRelatedItems ssm:ListOpsMetadata ssm:ListResourceComplianceSummaries ssm:ListResourceDataSync ssm:ModifyDocumentPermission ssm:PutComplianceItems ssm:PutInventory ssm:PutParameter ssm:PutResourcePolicy ssm:RegisterDefaultPatchBaseline ssm:RegisterManagedInstance ssm:RegisterPatchBaselineForPatchGroup ssm:RegisterTargetWithMaintenanceWindow ssm:RegisterTaskWithMaintenanceWindow ssm:ResetServiceSetting ssm:ResumeSession ssm:SendAutomationSignal ssm:SendCommand ssm:StartAssociationsOnce ssm:StartAutomationExecution ssm:StartChangeRequestExecution

服务前缀	操作
	ssm:StartSession ssm:StopAutomationExecution ssm:TerminateSession ssm:UnlabelParameterVersion ssm:UpdateAssociation ssm:UpdateAssociationStatus ssm:UpdateDocument ssm:UpdateDocumentDefaultVersion ssm:UpdateDocumentMetadata ssm:UpdateInstanceInformation ssm:UpdateMaintenanceWindow ssm:UpdateMaintenanceWindowTarget ssm:UpdateMaintenanceWindowTask ssm:UpdateManagedInstanceRole ssm:UpdateOpsItem ssm:UpdateOpsMetadata ssm:UpdatePatchBaseline ssm:UpdateResourceDataSync ssm:UpdateServiceSetting

服务前缀	操作
ssm-incidents	ssm-incidents:BatchGetIncidentFindings
	ssm-incidents:CreateReplicationSet
	ssm-incidents:CreateResponsePlan
	ssm-incidents:CreateTimelineEvent
	ssm-incidents:DeleteIncidentRecord
	ssm-incidents:DeleteReplicationSet
	ssm-incidents:DeleteResourcePolicy
	ssm-incidents:DeleteResponsePlan
	ssm-incidents:DeleteTimelineEvent
	ssm-incidents:GetIncidentRecord
	ssm-incidents:GetReplicationSet
	ssm-incidents:GetResourcePolicies
	ssm-incidents:GetResponsePlan
	ssm-incidents:GetTimelineEvent
	ssm-incidents:ListIncidentFindings
	ssm-incidents:ListIncidentRecords
	ssm-incidents:ListRelatedItems
	ssm-incidents:ListReplicationSets
	ssm-incidents:ListResponsePlans
	ssm-incidents:ListTimelineEvents
	ssm-incidents:PutResourcePolicy

服务前缀	操作
	ssm-incidents:StartIncident
	ssm-incidents:UpdateDeletionProtection
	ssm-incidents:UpdateIncidentRecord
	ssm-incidents:UpdateRelatedItems
	ssm-incidents:UpdateReplicationSet
	ssm-incidents:UpdateResponsePlan
	ssm-incidents:UpdateTimelineEvent

服务前缀	操作
ssm-sap	ssm-sap:BackupDatabase ssm-sap:DeleteResourcePermission ssm-sap:DeregisterApplication ssm-sap:GetApplication ssm-sap:GetComponent ssm-sap:GetConfigurationCheckOperation ssm-sap:GetDatabase ssm-sap:GetOperation ssm-sap:GetResourcePermission ssm-sap:ListApplications ssm-sap:ListComponents ssm-sap:ListConfigurationCheckDefinitions ssm-sap:ListConfigurationCheckOperations ssm-sap:ListDatabases ssm-sap:ListOperationEvents ssm-sap:ListOperations ssm-sap:ListSubCheckResults ssm-sap:ListSubCheckRuleResults ssm-sap:PutResourcePermission ssm-sap:RegisterApplication ssm-sap:RestoreDatabase

服务前缀	操作
	ssm-sap:StartApplication ssm-sap:StartApplicationRefresh ssm-sap:StartConfigurationChecks ssm-sap:StopApplication ssm-sap:UpdateApplicationSettings ssm-sap:UpdateHANABackupSettings

服务前缀	操作
states	states:CreateActivity
	states:CreateStateMachine
	states:CreateStateMachineAlias
	states:DeleteActivity
	states:DeleteStateMachine
	states:DeleteStateMachineAlias
	states:DeleteStateMachineVersion
	states:DescribeActivity
	states:DescribeExecution
	states:DescribeMapRun
	states:DescribeStateMachine
	states:DescribeStateMachineAlias
	states:DescribeStateMachineForExecution
	states:GetExecutionHistory
	states:ListActivities
	states:ListExecutions
	states:ListMapRuns
	states:ListStateMachineAliases
	states:ListStateMachineVersions
	states:ListStateMachines
	states:SendTaskFailure

服务前缀	操作
	states:SendTaskHeartbeat states:SendTaskSuccess states:StartExecution states:StopExecution states:UpdateMapRun states:UpdateStateMachine states:UpdateStateMachineAlias states:ValidateStateMachineDefinition
sts	sts:AssumeRole sts:AssumeRoleWithSAML sts:AssumeRoleWithWebIdentity sts:DecodeAuthorizationMessage sts:GetAccessKeyInfo sts:GetCallerIdentity sts:GetFederationToken sts:GetSessionToken sts:GetWebIdentityToken

服务前缀	操作
swf	swf:DeleteActivityType
	swf:DeleteWorkflowType
	swf:DeprecateActivityType
	swf:DeprecateDomain
	swf:DeprecateWorkflowType
	swf:DescribeActivityType
	swf:DescribeDomain
	swf:DescribeWorkflowType
	swf:ListActivityTypes
	swf:ListDomains
	swf:ListWorkflowTypes
	swf:RegisterActivityType
	swf:RegisterDomain
	swf:RegisterWorkflowType
	swf:UndeprecateActivityType
	swf:UndeprecateDomain
	swf:UndeprecateWorkflowType

服务前缀	操作
synthetics	synthetics:AssociateResource synthetics:CreateCanary synthetics:CreateGroup synthetics:DeleteCanary synthetics:DeleteGroup synthetics:DescribeCanaries synthetics:DescribeCanariesLastRun synthetics:DescribeRuntimeVersions synthetics:DisassociateResource synthetics:GetCanary synthetics:GetCanaryRuns synthetics:GetGroup synthetics:ListAssociatedGroups synthetics:ListGroupResources synthetics:ListGroups synthetics:StartCanary synthetics:StartCanaryDryRun synthetics:StopCanary synthetics:UpdateCanary

服务前缀	操作
标签	tag:DescribeReportCreation tag:GetComplianceSummary tag:GetResources tag:StartReportCreation

服务前缀	操作
textract	textract:AnalyzeDocument textract:AnalyzeExpense textract:AnalyzeID textract:CreateAdapter textract:CreateAdapterVersion textract:DeleteAdapter textract:DeleteAdapterVersion textract:DetectDocumentText textract:GetAdapter textract:GetAdapterVersion textract:GetDocumentAnalysis textract:GetDocumentTextDetection textract:GetExpenseAnalysis textract:GetLendingAnalysis textract:GetLendingAnalysisSummary textract:ListAdapterVersions textract:ListAdapters textract:StartDocumentAnalysis textract:StartDocumentTextDetection textract:StartExpenseAnalysis textract:StartLendingAnalysis

服务前缀	操作
	textract:UpdateAdapter

服务前缀	操作
timestream	timestream:CancelQuery timestream:CreateDatabase timestream:CreateScheduledQuery timestream:CreateTable timestream:DeleteDatabase timestream:DeleteScheduledQuery timestream:DeleteTable timestream:DescribeAccountSettings timestream:DescribeDatabase timestream:DescribeScheduledQuery timestream:DescribeTable timestream:ExecuteScheduledQuery timestream:ListBatchLoadTasks timestream:ListDatabases timestream:ListScheduledQueries timestream:ListTables timestream:PrepareQuery timestream:UpdateAccountSettings timestream:UpdateDatabase timestream:UpdateScheduledQuery timestream:UpdateTable

服务前缀	操作
tnb	tnb:CancelSolNetworkOperation tnb:CreateSolFunctionPackage tnb:CreateSolNetworkInstance tnb:CreateSolNetworkPackage tnb:DeleteSolFunctionPackage tnb:DeleteSolNetworkInstance tnb:DeleteSolNetworkPackage tnb:GetSolFunctionInstance tnb:GetSolFunctionPackage tnb:GetSolFunctionPackageContent tnb:GetSolFunctionPackageDescriptor tnb:GetSolNetworkInstance tnb:GetSolNetworkOperation tnb:GetSolNetworkPackage tnb:GetSolNetworkPackageContent tnb:GetSolNetworkPackageDescriptor tnb:InstantiateSolNetworkInstance tnb:ListSolFunctionInstances tnb:ListSolFunctionPackages tnb:ListSolNetworkInstances tnb:ListSolNetworkOperations

服务前缀	操作
	tnb:ListSolNetworkPackages tnb:PutSolFunctionPackageContent tnb:PutSolNetworkPackageContent tnb:TerminateSolNetworkInstance tnb:UpdateSolFunctionPackage tnb:UpdateSolNetworkInstance tnb:UpdateSolNetworkPackage tnb:ValidateSolFunctionPackageContent tnb:ValidateSolNetworkPackageContent

服务前缀	操作
transcribe	transcribe:CreateCallAnalyticsCategory transcribe:CreateLanguageModel transcribe:CreateMedicalVocabulary transcribe:CreateVocabulary transcribe:CreateVocabularyFilter transcribe:DeleteCallAnalyticsCategory transcribe:DeleteCallAnalyticsJob transcribe:DeleteLanguageModel transcribe:DeleteMedicalScribeJob transcribe:DeleteMedicalTranscriptionJob transcribe:DeleteMedicalVocabulary transcribe:DeleteTranscriptionJob transcribe:DeleteVocabulary transcribe:DeleteVocabularyFilter transcribe:DescribeLanguageModel transcribe:GetCallAnalyticsCategory transcribe:GetCallAnalyticsJob transcribe:GetMedicalScribeJob transcribe:GetMedicalTranscriptionJob transcribe:GetMedicalVocabulary transcribe:GetTranscriptionJob

服务前缀	操作
	transcribe:GetVocabulary
	transcribe:GetVocabularyFilter
	transcribe:ListCallAnalyticsCategories
	transcribe:ListCallAnalyticsJobs
	transcribe:ListLanguageModels
	transcribe:ListMedicalScribeJobs
	transcribe:ListMedicalTranscriptionJobs
	transcribe:ListMedicalVocabularies
	transcribe:ListTranscriptionJobs
	transcribe:ListVocabularies
	transcribe:ListVocabularyFilters
	transcribe:StartCallAnalyticsJob
	transcribe:StartCallAnalyticsStreamTranscription
	transcribe:StartCallAnalyticsStreamTranscriptionWebSocket
	transcribe:StartMedicalScribeJob
	transcribe:StartMedicalStreamTranscription
	transcribe:StartMedicalStreamTranscriptionWebSocket
	transcribe:StartMedicalTranscriptionJob
	transcribe:StartStreamTranscription
	transcribe:StartStreamTranscriptionWebSocket
	transcribe:StartTranscriptionJob

服务前缀	操作
	transcribe:UpdateCallAnalyticsCategory transcribe:UpdateMedicalVocabulary transcribe:UpdateVocabulary transcribe:UpdateVocabularyFilter

服务前缀	操作
转移	transfer:CreateAccess transfer:CreateAgreement transfer:CreateConnector transfer:CreateProfile transfer:CreateServer transfer:CreateUser transfer:CreateWebApp transfer:CreateWorkflow transfer:DeleteAccess transfer:DeleteAgreement transfer:DeleteCertificate transfer:DeleteConnector transfer:DeleteHostKey transfer:DeleteProfile transfer:DeleteServer transfer:DeleteSshPublicKey transfer:DeleteUser transfer:DeleteWebApp transfer:DeleteWebAppCustomization transfer:DeleteWorkflow transfer:DescribeAccess

服务前缀	操作
	transfer:DescribeAgreement transfer:DescribeCertificate transfer:DescribeConnector transfer:DescribeExecution transfer:DescribeHostKey transfer:DescribeProfile transfer:DescribeSecurityPolicy transfer:DescribeServer transfer:DescribeUser transfer:DescribeWebApp transfer:DescribeWebAppCustomization transfer:DescribeWorkflow transfer:ImportCertificate transfer:ImportHostKey transfer:ImportSshPublicKey transfer:ListAccesses transfer:ListCertificates transfer:ListConnectors transfer:ListExecutions transfer:ListFileTransferResults transfer:ListHostKeys

服务前缀	操作
	transfer:ListProfiles transfer:ListSecurityPolicies transfer:ListServers transfer:ListUsers transfer:ListWebApps transfer:ListWorkflows transfer:SendWorkflowStepState transfer:StartDirectoryListing transfer:StartFileTransfer transfer:StartRemoteDelete transfer:StartRemoteMove transfer:StartServer transfer:StopServer transfer:TestConnection transfer:TestIdentityProvider transfer:UpdateAccess transfer:UpdateAgreement transfer:UpdateCertificate transfer:UpdateConnector transfer:UpdateHostKey transfer:UpdateProfile

服务前缀	操作
	transfer:UpdateServer transfer:UpdateUser transfer:UpdateWebApp transfer:UpdateWebAppCustomization
翻译	translate:CreateParallelData translate:DeleteParallelData translate:DeleteTerminology translate:DescribeTextTranslationJob translate:GetParallelData translate:GetTerminology translate:ImportTerminology translate:ListLanguages translate:ListParallelData translate:ListTerminologies translate:ListTextTranslationJobs translate:StartTextTranslationJob translate:StopTextTranslationJob translate:TranslateDocument translate:TranslateText translate:UpdateParallelData

服务前缀	操作
voiceid	voiceid:AssociateFraudster voiceid:CreateDomain voiceid:CreateWatchlist voiceid:DeleteDomain voiceid:DeleteFraudster voiceid:DeleteSpeaker voiceid:DeleteWatchlist voiceid:DescribeDomain voiceid:DescribeFraudster voiceid:DescribeFraudsterRegistrationJob voiceid:DescribeSpeaker voiceid:DescribeSpeakerEnrollmentJob voiceid:DescribeWatchlist voiceid:DisassociateFraudster voiceid:EvaluateSession voiceid:ListDomains voiceid:ListFraudsterRegistrationJobs voiceid:ListFraudsters voiceid:ListSpeakerEnrollmentJobs voiceid:ListSpeakers voiceid:ListWatchlists

服务前缀	操作
	voiceid:OptOutSpeaker voiceid:StartFraudsterRegistrationJob voiceid:StartSpeakerEnrollmentJob voiceid:UpdateDomain voiceid:UpdateWatchlist

服务前缀	操作
vpc-lattice	vpc-lattice:CreateAccessLogSubscription
	vpc-lattice:CreateListener
	vpc-lattice:CreateResourceConfiguration
	vpc-lattice:CreateResourceGateway
	vpc-lattice:CreateRule
	vpc-lattice:CreateService
	vpc-lattice:CreateServiceNetwork
	vpc-lattice:CreateServiceNetworkResourceAssociation
	vpc-lattice:CreateServiceNetworkServiceAssociation
	vpc-lattice:CreateServiceNetworkVpcAssociation
	vpc-lattice:CreateTargetGroup
	vpc-lattice:DeleteAccessLogSubscription
	vpc-lattice:DeleteAuthPolicy
	vpc-lattice:DeleteDomainVerification
	vpc-lattice:DeleteListener
	vpc-lattice:DeleteResourceConfiguration
	vpc-lattice:DeleteResourceEndpointAssociation
	vpc-lattice:DeleteResourceGateway
	vpc-lattice:DeleteResourcePolicy
	vpc-lattice:DeleteRule
	vpc-lattice:DeleteService

服务前缀	操作
	vpc-lattice:DeleteServiceNetwork
	vpc-lattice:DeleteServiceNetworkResourceAssociation
	vpc-lattice:DeleteServiceNetworkServiceAssociation
	vpc-lattice:DeleteServiceNetworkVpcAssociation
	vpc-lattice:DeleteTargetGroup
	vpc-lattice:DeregisterTargets
	vpc-lattice:GetAccessLogSubscription
	vpc-lattice:GetAuthPolicy
	vpc-lattice:GetDomainVerification
	vpc-lattice:GetListener
	vpc-lattice:GetResourceConfiguration
	vpc-lattice:GetResourceGateway
	vpc-lattice:GetResourcePolicy
	vpc-lattice:GetRule
	vpc-lattice:GetService
	vpc-lattice:GetServiceNetwork
	vpc-lattice:GetServiceNetworkResourceAssociation
	vpc-lattice:GetServiceNetworkServiceAssociation
	vpc-lattice:GetServiceNetworkVpcAssociation
	vpc-lattice:GetTargetGroup
	vpc-lattice:ListAccessLogSubscriptions

服务前缀	操作
	vpc-lattice:ListDomainVerifications
	vpc-lattice:ListListeners
	vpc-lattice:ListResourceConfigurations
	vpc-lattice:ListResourceEndpointAssociations
	vpc-lattice:ListResourceGateways
	vpc-lattice:ListRules
	vpc-lattice:ListServiceNetworkResourceAssociations
	vpc-lattice:ListServiceNetworkServiceAssociations
	vpc-lattice:ListServiceNetworkVpcAssociations
	vpc-lattice:ListServiceNetworkVpcEndpointAssociations
	vpc-lattice:ListServiceNetworks
	vpc-lattice:ListServices
	vpc-lattice:ListTargetGroups
	vpc-lattice:ListTargets
	vpc-lattice:PutAuthPolicy
	vpc-lattice:PutResourcePolicy
	vpc-lattice:RegisterTargets
	vpc-lattice:StartDomainVerification
	vpc-lattice:UpdateAccessLogSubscription
	vpc-lattice:UpdateListener
	vpc-lattice:UpdateResourceConfiguration

服务前缀	操作
	vpc-lattice:UpdateResourceGateway
	vpc-lattice:UpdateRule
	vpc-lattice:UpdateService
	vpc-lattice:UpdateServiceNetwork
	vpc-lattice:UpdateServiceNetworkVpcAssociation
	vpc-lattice:UpdateTargetGroup

服务前缀	操作
wafv2	wafv2:AssociateWebACL wafv2:CheckCapacity wafv2:CreateAPIKey wafv2:CreateIPSet wafv2:CreateRegexPatternSet wafv2:CreateRuleGroup wafv2:CreateWebACL wafv2:DeleteAPIKey wafv2:DeleteFirewallManagerRuleGroups wafv2:DeleteIPSet wafv2:DeleteLoggingConfiguration wafv2:DeletePermissionPolicy wafv2:DeleteRegexPatternSet wafv2:DeleteRuleGroup wafv2:DeleteWebACL wafv2:DescribeAllManagedProducts wafv2:DescribeManagedProductsByVendor wafv2:DescribeManagedRuleGroup wafv2:DisassociateWebACL wafv2:GenerateMobileSdkReleaseUrl wafv2:GetDecryptedAPIKey

服务前缀	操作
	wafv2:GetIPSet wafv2:GetLoggingConfiguration wafv2:GetManagedRuleSet wafv2:GetMobileSdkRelease wafv2:GetRateBasedStatementManagedKeys wafv2:GetRegexPatternSet wafv2:GetRuleGroup wafv2:GetSampledRequests wafv2:GetWebACLForResource wafv2:ListAPIKeys wafv2:ListAvailableManagedRuleGroupVersions wafv2:ListAvailableManagedRuleGroups wafv2:ListIPSets wafv2:ListLoggingConfigurations wafv2:ListManagedRuleSets wafv2:ListMobileSdkReleases wafv2:ListRegexPatternSets wafv2:ListResourcesForWebACL wafv2:ListRuleGroups wafv2:ListWebACLs wafv2:PutLoggingConfiguration

服务前缀	操作
	wafv2:PutManagedRuleSetVersions wafv2:UpdateIPSet wafv2:UpdateManagedRuleSetVersionExpiryDate wafv2:UpdateRegexPatternSet wafv2:UpdateRuleGroup wafv2:UpdateWebACL

服务前缀	操作
wellarchitected	wellarchitected:AssociateLenses
	wellarchitected:AssociateProfiles
	wellarchitected:CreateLensShare
	wellarchitected:CreateLensVersion
	wellarchitected:CreateMilestone
	wellarchitected:CreateProfile
	wellarchitected:CreateProfileShare
	wellarchitected:CreateReviewTemplate
	wellarchitected:CreateWorkload
	wellarchitected:CreateWorkloadShare
	wellarchitected:DeleteLens
	wellarchitected:DeleteLensShare
	wellarchitected:DeleteProfile
	wellarchitected:DeleteProfileShare
	wellarchitected:DeleteReviewTemplate
	wellarchitected:DeleteTemplateShare
	wellarchitected:DeleteWorkload
	wellarchitected:DeleteWorkloadShare
	wellarchitected:DisassociateLenses
	wellarchitected:DisassociateProfiles
	wellarchitected:ExportLens

服务前缀	操作
	wellarchitected:GetAnswer
	wellarchitected:GetConsolidatedReport
	wellarchitected:GetGlobalSettings
	wellarchitected:GetLens
	wellarchitected:GetLensReview
	wellarchitected:GetLensReviewReport
	wellarchitected:GetLensVersionDifference
	wellarchitected:GetMilestone
	wellarchitected:GetProfile
	wellarchitected:GetProfileTemplate
	wellarchitected:GetReviewTemplate
	wellarchitected:GetReviewTemplateAnswer
	wellarchitected:GetReviewTemplateLensReview
	wellarchitected:GetWorkload
	wellarchitected:ImportLens
	wellarchitected:ListAnswers
	wellarchitected:ListCheckDetails
	wellarchitected:ListCheckSummaries
	wellarchitected:ListLensReviewImprovements
	wellarchitected:ListLensReviews
	wellarchitected:ListLensShares

服务前缀	操作
	wellarchitected:ListLenses
	wellarchitected:ListMilestones
	wellarchitected:ListNotifications
	wellarchitected:ListProfileNotifications
	wellarchitected:ListProfileShares
	wellarchitected:ListProfiles
	wellarchitected:ListReviewTemplateAnswers
	wellarchitected:ListReviewTemplates
	wellarchitected:ListShareInvitations
	wellarchitected:ListTemplateShares
	wellarchitected:ListWorkloadShares
	wellarchitected:ListWorkloads
	wellarchitected:UpdateAnswer
	wellarchitected:UpdateGlobalSettings
	wellarchitected:UpdateIntegration
	wellarchitected:UpdateLensReview
	wellarchitected:UpdateProfile
	wellarchitected:UpdateReviewTemplate
	wellarchitected:UpdateReviewTemplateLensReview
	wellarchitected:UpdateShareInvitation
	wellarchitected:UpdateWorkload

服务前缀	操作
	wellarchitected:UpdateWorkloadShare
	wellarchitected:UpgradeLensReview
	wellarchitected:UpgradeProfileVersion
	wellarchitected:UpgradeReviewTemplateLensReview

服务前缀	操作
wisdom	wisdom:CreateAssistant wisdom:CreateAssistantAssociation wisdom:CreateContent wisdom:CreateKnowledgeBase wisdom:CreateQuickResponse wisdom:CreateSession wisdom:DeleteAssistant wisdom:DeleteAssistantAssociation wisdom:DeleteContent wisdom:DeleteImportJob wisdom:DeleteKnowledgeBase wisdom:DeleteQuickResponse wisdom:GetAssistant wisdom:GetAssistantAssociation wisdom:GetContent wisdom:GetContentAssociation wisdom:GetContentSummary wisdom:GetImportJob wisdom:GetKnowledgeBase wisdom:GetRecommendations wisdom:GetSession

服务前缀	操作
	wisdom:ListAssistantAssociations wisdom:ListAssistants wisdom:ListContentAssociations wisdom:ListContents wisdom:ListImportJobs wisdom:ListKnowledgeBases wisdom:ListQuickResponses wisdom:NotifyRecommendationsReceived wisdom:QueryAssistant wisdom:RemoveKnowledgeBaseTemplateUri wisdom:SearchContent wisdom:SearchQuickResponses wisdom:SearchSessions wisdom:StartContentUpload wisdom:StartImportJob wisdom:UpdateContent wisdom:UpdateKnowledgeBaseTemplateUri wisdom:UpdateQuickResponse wisdom:UpdateSession

服务前缀	操作
worklink	worklink:AssociateDomain
	worklink:AssociateWebsiteAuthorizationProvider
	worklink:AssociateWebsiteCertificateAuthority
	worklink:CreateFleet
	worklink:DeleteFleet
	worklink:DescribeAuditStreamConfiguration
	worklink:DescribeCompanyNetworkConfiguration
	worklink:DescribeDevice
	worklink:DescribeDevicePolicyConfiguration
	worklink:DescribeDomain
	worklink:DescribeFleetMetadata
	worklink:DescribeIdentityProviderConfiguration
	worklink:DescribeWebsiteCertificateAuthority
	worklink:DisassociateDomain
	worklink:DisassociateWebsiteAuthorizationProvider
	worklink:DisassociateWebsiteCertificateAuthority
	worklink:ListDevices
	worklink:ListDomains
	worklink:ListFleets
	worklink:ListWebsiteAuthorizationProviders
	worklink:ListWebsiteCertificateAuthorities

服务前缀	操作
	worklink:RestoreDomainAccess
	worklink:RevokeDomainAccess
	worklink:SignOutUser
	worklink:UpdateAuditStreamConfiguration
	worklink:UpdateCompanyNetworkConfiguration
	worklink:UpdateDevicePolicyConfiguration
	worklink:UpdateDomainMetadata
	worklink:UpdateFleetMetadata
	worklink:UpdateIdentityProviderConfiguration

服务前缀	操作
工作区	workspaces:AcceptAccountLinkInvitation workspaces:AssociateConnectionAlias workspaces:AssociateIpGroups workspaces:AssociateWorkspaceApplication workspaces:CopyWorkspaceImage workspaces:CreateAccountLinkInvitation workspaces:CreateConnectClientAddIn workspaces:CreateConnectionAlias workspaces:CreateIpGroup workspaces:CreateStandbyWorkspaces workspaces:CreateUpdatedWorkspaceImage workspaces:CreateWorkspaceBundle workspaces:CreateWorkspaceImage workspaces:CreateWorkspaces workspaces:CreateWorkspacesPool workspaces:DeleteAccountLinkInvitation workspaces:DeleteClientBranding workspaces:DeleteConnectClientAddIn workspaces:DeleteConnectionAlias workspaces:DeleteIpGroup workspaces:DeleteWorkspaceBundle

服务前缀	操作
	workspaces:DeleteWorkspaceImage workspaces:DeployWorkspaceApplications workspaces:DeregisterWorkspaceDirectory workspaces:DescribeAccount workspaces:DescribeAccountModifications workspaces:DescribeApplicationAssociations workspaces:DescribeApplications workspaces:DescribeBundleAssociations workspaces:DescribeClientBranding workspaces:DescribeClientProperties workspaces:DescribeConnectClientAddIns workspaces:DescribeConnectionAliasPermissions workspaces:DescribeConnectionAliases workspaces:DescribeCustomWorkspaceImageImport workspaces:DescribeImageAssociations workspaces:DescribeIpGroups workspaces:DescribeWorkspaceAssociations workspaces:DescribeWorkspaceBundles workspaces:DescribeWorkspaceDirectories workspaces:DescribeWorkspaceImagePermissions workspaces:DescribeWorkspaceSnapshots

服务前缀	操作
	workspaces:DescribeWorkspaces workspaces:DescribeWorkspacesConnectionStatus workspaces:DescribeWorkspacesPoolSessions workspaces:DescribeWorkspacesPools workspaces:DisassociateConnectionAlias workspaces:DisassociateIpGroups workspaces:DisassociateWorkspaceApplication workspaces:GetAccountLink workspaces:ImportClientBranding workspaces:ImportWorkspaceImage workspaces:ListAccountLinks workspaces:ListAvailableManagementCidrRanges workspaces:MigrateWorkspace workspaces:ModifyAccount workspaces:ModifyCertificateBasedAuthProperties workspaces:ModifyClientProperties workspaces:ModifyEndpointEncryptionMode workspaces:ModifySamlProperties workspaces:ModifySelfservicePermissions workspaces:ModifyStreamingProperties workspaces:ModifyWorkspaceAccessProperties

服务前缀	操作
	workspaces:ModifyWorkspaceCreationProperties workspaces:ModifyWorkspaceProperties workspaces:ModifyWorkspaceState workspaces:RebootWorkspaces workspaces:RebuildWorkspaces workspaces:RegisterWorkspaceDirectory workspaces:RejectAccountLinkInvitation workspaces:RestoreWorkspace workspaces:StartWorkspaces workspaces:StartWorkspacesPool workspaces:StopWorkspaces workspaces:StopWorkspacesPool workspaces:TerminateWorkspaces workspaces:TerminateWorkspacesPool workspaces:TerminateWorkspacesPoolSession workspaces:UpdateConnectClientAddIn workspaces:UpdateConnectionAliasPermission workspaces:UpdateWorkspaceBundle workspaces:UpdateWorkspaceImagePermission workspaces:UpdateWorkspacesPool

服务前缀	操作
xray	xray:CreateGroup xray:CreateSamplingRule xray:DeleteGroup xray:DeleteResourcePolicy xray:DeleteSamplingRule xray:GetEncryptionConfig xray:GetGroup xray:GetGroups xray:GetInsight xray:GetInsightEvents xray:GetInsightImpactGraph xray:GetInsightSummaries xray:GetSamplingRules xray:ListResourcePolicies xray:PutEncryptionConfig xray:PutResourcePolicy xray:UpdateGroup xray:UpdateSamplingRule

IAM Access Analyzer 配额

IAM Access Analyzer 具有以下配额：

资源	默认配额	最大配额
每个区域每个 Amazon Web Services 账户 每个分析器类型的最大账户级别分析器数量	1	1
每个区域每个 Amazon Web Services 账户每个分析器类型的最大组织级别外部或未使用访问分析器数量	5	20[1]
每个区域每个 Amazon 组织的最大组织级别内部访问分析器数量	1	1
每个分析器的最大存档规则数	100 对于每个标准，每个存档规则可具有最多 20 个值。	1,000[1]
每个分析器每小时的最大访问预览数	1000	1000
Amazon CloudTrail每个策略生成处理的 日志文件	100000	100000
并发策略生成器	1	1
策略生成 Amazon CloudTrail 数据大小	25 GB	25 GB
策略生成 Amazon CloudTrail 时间范围	90 天	90 天
每天生成策略	非洲（开普敦）：5 亚太地区（香港）：5 亚太地区（雅加达）：5	非洲（开普敦）：5 亚太地区（香港）：5 亚太地区（雅加达）：5

资源	默认配额	最大配额
	欧洲（米兰）：5 中东（巴林）：5 所有其他支持的区域：50 **Note** 已取消的策略生成请求适用于每日配额。	欧洲（米兰）：5 中东（巴林）：5 所有其他支持的区域：50

[1]某些配额可由客户通过 Service Quotas 进行配置。

排查 IAM 问题

使用此处的信息可帮助您诊断和修复在使用 Amazon Identity and Access Management（IAM）时出现的常见问题。

问题

- 我无法登录我的 Amazon 账户
- 我丢失了访问密钥
- 策略变量不起作用
- 我所做的更改可能不会立即可见
- 我没有权限执行：iam:DeleteVirtualMFADevice
- 如何安全地创建 IAM 用户？
- 其他资源
- 排查访问被拒绝错误消息
- 排查根用户问题
- 排查 IAM 策略问题
- 排查通行密钥和 FIDO 安全密钥问题
- 排查 IAM 角色问题
- 排查 IAM 和 Amazon EC2 问题
- 排查 IAM 和 Amazon S3 问题
- 排查 SAML 与 IAM 联合身份验证的问题

我无法登录我的 Amazon 账户

确认您具有正确的凭证，并且使用正确的方法进行登录。有关更多信息，请参阅《Amazon 登录 用户指南》中的登录问题排查。

我丢失了访问密钥

访问密钥由两个部分组成：

- 访问密钥标识符。标识符是公开的，您可以在列出访问密钥的任意 IAM 控制台中进行查看，例如用户摘要页面。

- 秘密访问密钥。该密钥会在您最初创建访问密钥对时提供。它与密码一样，之后无法检索。如果您丢失了私有访问密钥，则必须创建新的访问密钥对。如果您已达到访问密钥的数量上限，必须删除一个现有的密钥对，然后才可以创建另一个。

如果您丢失了秘密访问密钥，则必须删除访问密钥并创建新的访问密钥。有关更多说明，请参阅 更新访问密钥。

策略变量不起作用

如果您的策略变量不起作用，则已发生下列错误之一：

版本策略元素中的日期错误。

验证包含变量的所有策略（包括以下策略版本编号）："Version": "2012-10-17"。如果没有正确的版本编号，则在评估过程中不会更换变量。只会从字面上评估这些变量。当您包含最新的版本号时，不包含变量的任何策略仍然有效。

Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。当您修改 IAM 中的客户管理型策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅IAM JSON 策略元素：Version。要了解策略版本的更多信息，请参阅the section called “IAM 策略版本控制”。

变量字符的字母大小写错误。

验证您的策略变量为正确的大小写。有关更多信息，请参阅 IAM policy 元素：变量和标签。

我所做的更改可能不会立即可见

作为全球数据中心的计算机要访问的服务，IAM 使用称为最终一致性的分布式计算模型。您在 IAM（或其他 Amazon 服务）中所做的任何更改，包括基于属性的访问权限控制（ABAC）标签，都需要一段时间才能在所有可能的端点中可见。它在服务器与服务器之间、复制区域与复制区域之间，以及区域与区域之间发送数据需要时间，这会造成一些延迟。IAM 也使用缓存来提高性能，但在某些情况下，这可能会增加时间。在之前缓存的数据超时之前，更改可能不可见。

您在设计全球应用程序时，必须考虑到这些可能的延迟。确保应用程序可以按预期工作，即使在一个位置进行的更改不能立即在其他位置可见。此类更改包括创建或更新用户、组、角色或策略。在应用程序的关键、高可用性代码路径中，我们不建议包含此类 IAM 更改。而应在不常运行的、单独的初始化或设置例程中进行 IAM 更改。另外，在生产工作流程依赖这些更改之前，请务必验证更改已传播。

有关其他某些 Amazon 服务如何受此影响的更多信息，请参阅以下资源：

- Amazon DynamoDB：《DynamoDB 开发人员指南》中的读取一致性，以及《Amazon DynamoDB 开发人员指南》中的读取一致性。
- Amazon EC2：Amazon EC2 API 参考中的 EC2 最终一致性。
- Amazon EMR：Amazon 大数据博客中的 Ensuring Consistency When Using Amazon S3 and Amazon EMR for ETL Workflows
- Amazon Redshift：Amazon Redshift 数据库开发人员指南中的管理数据一致性
- Amazon S3：Amazon Simple Storage Service 用户指南中的 Amazon S3 数据一致性模型

我没有权限执行：iam:DeleteVirtualMFADevice

当您尝试为自己或其他用户分配或删除虚拟 MFA 设备时，可能会收到以下错误：

```
User: arn:aws:iam::123456789012:user/Diego is not authorized to
 perform: iam:DeleteVirtualMFADevice on resource: arn:aws:iam::123456789012:mfa/Diego
 with an explicit deny
```

如果某个人以前在 IAM 控制台中开始将虚拟 MFA 设备分配给用户后取消该过程，则可能会发生这种情况。这会在 IAM 中为用户创建一个虚拟 MFA 设备，但绝不将其分配给用户。删除现有的虚拟 MFA 设备，然后使用相同设备名称创建新的虚拟 MFA 设备。

要修复此问题，管理员不应编辑策略权限。相反，管理员必须使用 Amazon CLI 或 Amazon API 来删除现有但未分配的虚拟 MFA 设备。

删除现有但未分配的虚拟 MFA 设备

1. 查看您账户中的虚拟 MFA 设备。
   - Amazon CLI: `aws iam list-virtual-mfa-devices`
   - Amazon API：`ListVirtualMFADevices`
2. 在响应中，找到您要修复的用户的虚拟 MFA 设备 ARN。
3. 删除虚拟 MFA 设备。
   - Amazon CLI: `aws iam delete-virtual-mfa-device`
   - Amazon API：`DeleteVirtualMFADevice`

如何安全地创建 IAM 用户？

如果您的员工需要访问 Amazon，则可以选择创建 IAM 用户或使用 IAM Identity Center 进行身份验证。如果使用 IAM，Amazon 建议您创建 IAM 用户并将凭证安全地传达给员工。如果您不在员工身旁，请使用安全的工作流程向员工传达凭证。

使用以下安全工作流程在 IAM 中创建新用户：

1. 使用 Amazon Web Services 管理控制台 创建新用户。选择使用生成的密码授予 Amazon Web Services 管理控制台 访问权限。如有必要，请选中 Users must create a new password at next sign-in（用户必须在下次登录时创建新密码）复选框。在用户更改密码之前，不要向用户添加权限策略。
2. 添加用户后，复制新用户的登录 URL、用户名和密码。要查看密码，请选择 Show（显示）。
3. 使用贵公司中的安全通信方法（如电子邮件、聊天或支持工单系统）将密码发送给员工。单独为您的用户提供 IAM 用户控制台链接及其用户名。在授予员工权限之前，告诉员工确认他们可以成功登录。
4. 员工确认后，添加他们所需的权限。作为安全最佳实践，添加一个策略，要求用户使用 MFA 进行身份验证以管理其凭证。有关策略示例，请参阅 Amazon：允许使用 MFA 完成身份验证的 IAM 用户在“安全凭证”页面上管理自己的凭证。。

其他资源

下列相关资源有助于您在使用 Amazon 期间排查问题。

- Amazon CloudTrail 用户指南 - 使用 Amazon CloudTrail 可跟踪对 Amazon 进行的 API 调用的历史记录并将这些信息存储在日志文件中。这有助于确定访问了您账户中的资源的用户和账户、进行调用的时间、请求的操作等。有关更多信息，请参阅 使用 Amazon CloudTrail 记录 IAM 和 Amazon STS API 调用。
- Amazon 知识中心 – 查找常见问题解答和其他资源的链接，帮助您排查问题。
- Amazon 支持中心 – 获取技术支持。
- Amazon Premium Support 中心 – 获取高级技术支持。

排查访问被拒绝错误消息

以下信息可以帮助您识别、诊断和解决 Amazon Identity and Access Management 访问被拒绝错误。当 Amazon 显式或隐式拒绝授权请求时，将显示拒绝访问错误讯息。

- 当策略包含特定的 Amazon 操作的 Deny 语句时，将发生显式拒绝。
- 当没有适用的 Deny 语句且没有适用的 Allow 语句时，会发生隐式拒绝。由于 IAM policy 默认拒绝 IAM 主体，因此该策略必须显式允许主体执行操作。否则，该策略会隐式拒绝访问。有关更多信息，请参阅 显式拒绝和隐式拒绝之间的区别。

当您向服务或资源发出请求时，可能会有多个策略适用于该请求。除了错误消息中指定的策略外，还要查看所有适用的策略。

- 如果同一策略类型的多个策略拒绝某个请求，则访问被拒绝错误消息不会指定评估的策略数量。
- 如果授权请求被多种策略类型拒绝，则 Amazon 仅在错误消息中包含其中一种策略类型。

> **Important**
>
> 登录 Amazon 时遇到问题？ 请确保使用的是适合您用户类型的 Amazon 登录页面。如果您是 Amazon Web Services 账户根用户（账户拥有者），则可以使用您在创建 Amazon Web Services 账户 时设置的凭证登录 Amazon。如果您是 IAM 用户，则您的账户管理员可以向您提供 Amazon 登录凭证。如果您需要请求支持，则请不要使用此页面上的反馈链接。表格由 Amazon 文档团队接收，而不是 Amazon Web Services 支持。请在联系我们页面上选择仍然无法登录 Amazon 账户，然后选择一个可用的支持选项。

当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝”

- 检查错误消息是否包含负责拒绝访问的策略类型和 Amazon 资源名称（ARN）。如果是这种情况，请检查指定策略中是否有针对该操作的拒绝语句。如果提供了策略类型但没有策略 ARN，则重点解决该策略类型的问题：检查指定类型的策略中是否有针对该操作的拒绝语句。如果错误消息没有提到与拒绝访问有关的策略类型，请使用本部分中的其余指南进一步排除故障。
- 验证您是否具有调用您请求的操作和资源的基于身份的策略权限。如果设置了任何条件，您还必须在发送请求时满足这些条件。有关查看或修改用于 IAM 用户、组或角色的策略的信息，请参阅 管理 IAM 策略。
- 如果 Amazon Web Services 管理控制台 返回消息称您无权执行某个操作，则必须联系您的管理员寻求帮助。您的管理员为您提供了登录凭证或登录链接。

  当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不具有虚构 `widgets:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 widgets:GetWidget on resource: my-example-widget
```

在这种情况下，Mateo 必须请求他的管理员更新其策略，以允许他使用 `widgets:GetWidget` 操作访问 *my-example-widget* 资源。

- 您是否尝试访问支持基于资源的策略（如 Amazon S3、Amazon SNS 或 Amazon SQS）的服务？如果是，请确认策略将您指定为主体并为您授予访问权限。如果您在您的账户中向服务发出请求，基于身份的策略或基于资源的策略将向您授予相应的权限。如果您在不同的账户中向服务发出请求，则基于身份的策略和基于资源的策略这两者都必须向您授予权限。要查看哪些服务支持基于资源的策略，请参阅使用 IAM 的 Amazon 服务。
- 如果您的策略包含的条件具有键值对，请仔细检查。示例包括 `aws:RequestTag/tag-key` 全局条件密钥、 Amazon KMS `kms:EncryptionContext:encryption_context_key` 和多个服务支持的 `ResourceTag/tag-key` 条件键。确保键名称不与多个结果匹配。由于条件键名称不区分大小写，因为检查名为 `foo` 的键的条件将与 `foo`、`Foo` 或 `F00` 匹配。如果您的请求包含多个键值对，其中的键名称只是大小写形式不同，则您的访问可能会被意外拒绝。有关更多信息，请参阅 IAM JSON 策略元素：Condition。
- 如果您具有权限边界，请验证用于权限边界的策略是否允许您的请求。如果基于身份的策略允许请求，但权限边界不允许，则会拒绝请求。权限边界控制 IAM 主体（用户或角色）可以拥有的最大权限。基于资源的策略不受权限边界限制。权限边界不常用。有关 Amazon 如何评估这些策略的更多信息，请参阅 策略评估逻辑。
- 如果您手动签署请求（不使用 Amazon SDK），请验证您已正确签署请求。
- 如果您使用的是 Amazon VPC 端点策略，但出现了 Amazon CloudTrail 未记录的拒绝访问错误，则可能是因为 VPC 端点所有者账户与调用账户或目标角色账户不一致所致。

当我使用临时安全凭证发送请求时，收到了“访问被拒绝”

- 首先，请确保您未因与您的临时凭证无关的原因而被拒绝访问。有关更多信息，请参阅 当我向某个 Amazon 服务发送请求时，收到了“访问被拒绝”。
- 要验证服务是否接受临时安全凭证，请参阅使用 IAM 的 Amazon 服务。
- 验证您的请求是否采用了正确的签名和适当的格式。有关详细信息，请参阅工具包文档或 将临时凭证用于 Amazon 资源。
- 验证您的临时安全凭证没有过期。有关更多信息，请参阅 IAM 临时安全凭证。

- 验证 IAM 用户或角色拥有正确的权限。临时安全凭证的权限派生自 IAM 用户或角色。因此，权限限于向您已担任其临时凭证的角色授予的权限。有关如何确定临时安全凭证的权限的更多信息，请参阅临时安全凭证的权限。
- 如果您担任一个角色，您的角色会话可能受会话策略的限制。以编程方式使用 Amazon STS 请求临时安全凭证时，您可以选择传递内联或托管会话策略。会话策略是高级策略，在以编程方式为角色创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 `Policy` 参数传递单个 JSON 内联会话策略文档。您可以使用 `PolicyArns` 参数指定最多 10 个托管会话策略。生成的会话的权限是角色的基于身份的策略与会话策略的交集。或者，如果您的管理员或自定义程序为您提供临时凭证，它们可能已包含会话策略以限制您的访问。
- 如果您是 Amazon STS 联合用户主体，您的会话可能受会话策略的限制。您以 IAM 用户身份登录到 Amazon，然后请求联合令牌，以创建联合用户。有关更多信息，请参阅 通过自定义身份凭证代理程序请求凭证。在请求联合令牌时，如果您或您的身份代理传递了会话策略，则您的会话受这些策略的限制。生成的会话的权限是您的 IAM 用户的基于身份的策略与会话策略的交集。有关会话策略的更多信息，请参阅会话策略。
- 如果您使用角色访问具有基于资源的策略的资源，则验证策略已授予该角色权限。例如，以下策略允许 `MyRole` 从账户 `111122223333` 访问 `amzn-s3-demo-bucket`。

  JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Sid": "S3BucketPolicy",
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::111122223333:role/MyRole"]},
    "Action": ["s3:PutObject"],
    "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/*"]
  }]
}
```

拒绝访问错误消息示例

大多数拒绝访问的错误消息都以 `User` *`user`* `is not authorized to perform` *`action`* `on` *`resource`* `because` *`context`* 格式显示。在此示例中，*##*是被拒绝访问的主体的 ARN，*##*是策略拒绝的服务操作，而*##*是策略对其执行操作的资源的 ARN。*###*字段提供有关拒绝访问的策略类型的其他上下文。在某些情况下，它还包含拒绝访问策略的 ARN。

当策略因包含 Deny 语句而显式拒绝访问时，Amazon 将在拒绝访问错误消息中包含短语 `with an explicit deny in a type policy`。此短语还可以指定策略的 ARN，如下所示：`with an explicit deny in a type policy: policy ARN`。

当策略隐式拒绝访问时，Amazon 将在拒绝访问错误消息中包含短语 `because no type policy allows the action action`。

> **Note**
>
> 某些 Amazon 服务不支持这种拒绝访问的错误消息格式。拒绝访问错误消息的内容可能因发出授权请求的服务而异。

下面的示例展示了不同类型的拒绝访问错误消息的格式。

由于服务控制策略而拒绝访问 – 隐式拒绝

1. 检查服务控制策略（SCP）中的操作是否有缺少的 `Allow` 语句。对于以下示例，操作为 `codecommit:ListRepositories`。
2. 通过添加 `Allow` 语句来更新 SCP。有关更多信息，请参阅 Amazon Organizations 用户指南中的更新 SCP。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no service control policy allows the codecommit:ListRepositories action
```

由于服务控制策略而拒绝访问 – 显式拒绝

1. 如果错误消息中提供了策略 ARN，请查看指定服务控制策略（SCP）中操作的 `Deny` 语句。在下面的示例中，操作为 `codecommit:ListRepositories`。
2. 如果错误消息中未提供策略 ARN，请查看 SCP 中操作的 `Deny` 语句。
3. 通过删除 `Deny` 语句来更新 SCP。有关更多信息，请参阅《Amazon Organizations User Guide》中的 Update a service control policy (SCP)。

包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
with an explicit deny in a service control policy:
 arn:aws:organizations::777788889999:policy/o-exampleorgid/service_control_policy/p-
examplepolicyid123
```

不包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
with an explicit deny in a service control policy
```

由于资源控制策略而拒绝访问 – 显式拒绝

1. 如果错误消息中提供了策略 ARN，请查看指定资源控制策略（RCP）中操作的 Deny 语句。在下面的示例中，操作为 `secretsmanager:GetSecretValue`。
2. 如果错误消息中未提供策略 ARN，请查看 RCP 中操作的 Deny 语句。
3. 通过删除 Deny 语句来更新 RCP。有关更多信息，请参阅《Amazon Organizations User Guide》中的 Update a resource control policy (RCP)。

包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource control policy:
 arn:aws:organizations::777788889999:policy/o-exampleorgid/resource_control_policy/p-
examplepolicyid456
```

不包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource control policy
```

由于 VPC 端点策略而拒绝访问 – 隐式拒绝

1. 检查虚拟私有云（VPC）端点策略中的操作是否有缺失的 `Allow` 语句。对于以下示例，操作为 `codecommit:ListRepositories`。
2. 通过添加 `Allow` 语句来更新 VPC 端点策略。有关更多信息，请参阅《Amazon PrivateLink 指南》中的更新 VPC 端点策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no VPC endpoint policy allows the codecommit:ListRepositories action
```

由于 VPC 端点策略而拒绝访问 – 显式拒绝

1. 检查虚拟私有云（VPC）端点策略中的操作是否有显式 `Deny` 语句。对于以下示例，操作为 `codedeploy:ListDeployments`。
2. 通过删除 `Deny` 语句来更新 VPC 端点策略。有关更多信息，请参阅《Amazon PrivateLink 指南》中的更新 VPC 端点策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in a VPC endpoint policy
```

由于权限边界而拒绝访问 – 隐式拒绝

1. 检查权限边界中的操作是否有缺失的 `Allow` 语句。对于以下示例，操作为 `codedeploy:ListDeployments`。
2. 通过将 `Allow` 语句添加到 IAM policy 来更新权限边界。有关更多信息，请参阅IAM 实体的权限边界和编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
because no permissions boundary allows the codedeploy:ListDeployments action
```

由于权限边界而拒绝访问 – 显式拒绝

1. 如果错误消息中提供了策略 ARN，请查看指定权限边界中操作的 Deny 语句。在下面的示例中，操作为 `sagemaker:ListModels`。
2. 如果错误消息中提供了策略 ARN，请查看附加到主体的权限边界中操作的 Deny 语句。
3. 通过从 IAM policy 删除 Deny 语句来更新权限边界。有关更多信息，请参阅IAM 实体的权限边界和编辑 IAM 策略。

包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 sagemaker:ListModels
with an explicit deny in a permissions boundary: arn:aws:iam::123456789012:policy/
DeveloperPermissionBoundary
```

不包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 sagemaker:ListModels
with an explicit deny in a permissions boundary
```

由于会话策略而拒绝访问 – 隐式拒绝

1. 检查会话策略中的操作是否有缺失的 `Allow` 语句。对于以下示例，操作为 `codecommit:ListRepositories`。
2. 通过添加 `Allow` 语句来更新会话策略。有关更多信息，请参阅会话策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codecommit:ListRepositories
because no session policy allows the codecommit:ListRepositories action
```

由于会话策略而拒绝访问 – 显式拒绝

1. 如果错误消息中提供了策略 ARN，请查看指定会话策略中操作的 Deny 语句。在下面的示例中，操作为 `codedeploy:ListDeployments`。
2. 如果错误消息中未提供策略 ARN，请查看会话策略中操作的 Deny 语句。

3. 通过删除 Deny 语句来更新会话策略。有关更多信息，请参阅会话策略和 编辑 IAM 策略。

包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in a session policy: arn:aws:iam::123456789012:policy/
DeveloperSessionPolicy
```

不包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in a session policy
```

由于基于资源的策略而拒绝访问 – 隐式拒绝

1. 检查基于资源的策略中的操作是否有缺失的 Allow 语句。对于以下示例，操作为 secretsmanager:GetSecretValue。
2. 通过添加 Allow 语句来更新策略。有关更多信息，请参阅基于资源的策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
because no resource-based policy allows the secretsmanager:GetSecretValue action
```

由于基于资源的策略而拒绝访问 – 显式拒绝

1. 检查基于资源的策略中的操作是否有显式 Deny 语句。对于以下示例，操作为 secretsmanager:GetSecretValue。
2. 通过删除 Deny 语句来更新策略。有关更多信息，请参阅基于资源的策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform:
 secretsmanager:GetSecretValue
on resource: arn:aws:secretsmanager:us-east-1:123456789012:secret:*
with an explicit deny in a resource-based policy
```

由于角色信任策略而拒绝访问 – 隐式拒绝

1. 检查角色信任策略中的操作是否有缺失的 Allow 语句。对于以下示例，操作为 sts:AssumeRole。
2. 通过添加 Allow 语句来更新策略。有关更多信息，请参阅基于资源的策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform: sts:AssumeRole
because no role trust policy allows the sts:AssumeRole action
```

由于角色信任策略而拒绝访问 – 显式拒绝

1. 检查角色信任策略中的操作是否有显式 Deny 语句。对于以下示例，操作为 sts:AssumeRole。
2. 通过删除 Deny 语句来更新策略。有关更多信息，请参阅基于资源的策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:user/John is not authorized to perform: sts:AssumeRole
with an explicit deny in the role trust policy
```

由于基于身份的策略而拒绝访问 – 隐式拒绝

1. 检查附加到身份的基于身份的策略中的操作中是否有缺失的 Allow 语句。对于以下示例，操作为附加到角色 HR 的 codecommit:ListRepositories。
2. 通过添加 Allow 语句来更新策略。有关更多信息，请参阅基于身份的策略和 编辑 IAM 策略。

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codecommit:ListRepositories
because no identity-based policy allows the codecommit:ListRepositories action
```

由于基于身份的策略而拒绝访问 – 显式拒绝

1. 如果错误消息中提供了策略 ARN，请查看指定策略中操作的 Deny 语句。在下面的示例中，操作为 codedeploy:ListDeployments。
2. 如果错误消息中未提供策略 ARN，请查看附加到身份的、基于身份的策略中操作的 Deny 语句。
3. 通过删除 Deny 语句来更新策略。有关更多信息，请参阅基于身份的策略和 编辑 IAM 策略。

包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in an identity-based policy: arn:aws:iam::123456789012:policy/
HRAccessPolicy
```

不包含策略 ARN 的错误消息：

```
User: arn:aws:iam::123456789012:role/HR is not authorized to perform:
 codedeploy:ListDeployments
on resource: arn:aws:codedeploy:us-east-1:123456789012:deploymentgroup:*
with an explicit deny in an identity-based policy
```

排查根用户问题

使用此处的信息有助于排查与 Amazon Web Services 账户根用户相关的问题。

> **Note**
>
> 使用 Amazon Organizations 管理的 Amazon Web Services 账户可能为成员账户启用了集中的根访问。这些成员账户没有根用户凭证，无法以根用户身份登录，并且无法恢复根用户密码。如果您需要执行需要根用户凭证的任务，则请联系您的管理员。

在以账户根用户身份登录后，我无法执行预期能够完成的任务

您的账户可能是 Amazon Organizations 中组织的成员。组织管理员可能有服务控制策略（SCP）以限制您账户的权限。SCP 会影响根用户（包括根用户）。有关更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。

我忘记了我 Amazon Web Services 账户 的根用户密码

如果您是根用户并且丢失或忘记了 Amazon Web Services 账户的密码，则可以重置密码。您必须知道用于创建 Amazon Web Services 账户的电子邮件地址，并且必须有权访问该电子邮件账户。有关更多信息，请参阅 重置丢失或遗忘的根用户密码。

我无权访问我 Amazon Web Services 账户 的电子邮件

创建 Amazon Web Services 账户 时，您会提供电子邮件地址和密码。这些是 Amazon Web Services 账户根用户 的凭证。如果您不确定哪个电子邮件地址与您的 Amazon Web Services 账户关联，请搜

索从 @signin.aws 或 @verify.signin.aws 发往您组织中可能曾用于创建 Amazon Web Services 账户的电子邮件地址的邮件。

如果您知道电子邮件地址，但无法再访问电子邮件，请尝试恢复对电子邮件的访问权限。使用以下选项之一重新获得对您电子邮件的访问权限：

- 如果您拥有该电子邮件地址的域，则可以恢复已删除的电子邮件地址。或者，您可以为您的电子邮件账户设置捕获全部。“全部捕获”会收集发送到邮件服务器中不再存在的电子邮件地址的所有邮件，并将其重定向到另一个电子邮件地址。
- 如果账户上的电子邮件地址属于您的公司电子邮件系统，我们建议您联系 IT 系统管理员。它们也许能够帮助您重新获得电子邮件的访问权限。

如果仍然无法登录您的 Amazon Web Services 账户，您可以在 联系我们 中找到备选支持选项。

排查 IAM 策略问题

策略是 Amazon 中的一个实体；在附加到身份或资源时，策略定义它们的权限。在主体（如用户）发出请求时，Amazon 将评估这些策略。策略中的权限确定是允许还是拒绝请求。策略作为 JSON 文档存储在 Amazon 中，它们作为基于身份的策略 附加到主体或作为基于资源的策略 附加到资源。您可以将基于身份的策略附加到主体（或身份），例如，IAM 组、用户或角色。基于身份的策略包括 Amazon 托管策略、客户托管策略和内联策略。您可以在 Amazon Web Services 管理控制台 中使用可视化和 JSON 编辑器选项创建和编辑客户管理型策略。在 Amazon Web Services 管理控制台中查看策略时，您可以查看该策略授予的权限的摘要。您可以使用可视化编辑器和策略摘要帮助诊断和修复在管理 IAM policy 时遇到的常见错误。

请记住，所有 IAM policy 是使用以 JavaScript 对象表示法 (JSON) 规则开头的语法存储的。您无需了解该语法，即可创建或管理您的策略。您可以在 Amazon Web Services 管理控制台中使用可视化编辑器创建和编辑策略。要了解 IAM policy 中的 JSON 语法的更多信息，请参阅。IAM JSON 策略语言的语法

IAM policy 故障排除主题

- 使用可视化编辑器排除故障
  - 调整策略结构
  - 在可视化编辑器中选择资源 ARN
  - 在可视化编辑器中拒绝权限
  - 在可视化编辑器中指定多个服务

  - 在可视化编辑器中减小策略大小
  - 在可视化编辑器中修复无法识别的服务、操作或资源类型
- 排查策略摘要问题
  - 缺少策略摘要
  - 摘要策略包含无法识别的服务、操作或资源类型
  - 服务不支持 IAM policy 摘要
  - 我的策略未授予预期权限
- 策略管理故障排查
  - 在 IAM 账户中附加或分离策略
  - 根据您的 IAM 身份活动更改这些身份的策略
- JSON 策略文档故障排查
  - 验证您的策略
  - 我在 JSON 编辑器中没有用于策略验证的权限
  - 多个 JSON 策略对象
  - 多个 JSON Statement 元素
  - 在 JSON Statement 元素中具有多个 Effect、Action 或 Resource 元素
  - 缺少 JSON Version 元素

使用可视化编辑器排除故障

在创建或编辑客户管理型策略时，您可以使用可视化编辑器中的信息帮助纠正策略中的错误。要查看使用可视化编辑器创建策略的示例，请参阅the section called “控制对身份的访问”。

调整策略结构

在创建策略时，Amazon 将在存储之前验证、处理和转换该策略。检索策略后，Amazon 无需更改权限即可将其转换回人类可读的格式。这可能会导致您在策略可视化编辑器或 JSON 选项卡中看到的内容有所不同。

- 可以添加、删除或重新排序可视化编辑器权限块，并且可以优化块内的内容。
- 在 JSON 选项卡中，可以删除不太重要的空格，并且可以将 JSON 映射中的元素重新排序。此外，可以将主体元素中的 Amazon Web Services 账户 ID 替换为 Amazon Web Services 账户根用户 的 Amazon 资源名称（ARN）。

由于可能发生这些更改，不应以字符串形式来比较 JSON 策略文档。

在 Amazon Web Services 管理控制台 中创建客户管理型策略时，您可以选择完全在 JSON 编辑器中工作。如果从未在可视化编辑器中更改策略，并未从 JSON 选项卡中选择下一步，则不太可能会调整策略结构。当您使用可视化编辑器时，IAM 可能会重构策略以优化其外观。该结构调整仅在您的编辑会话中进行，不会自动进行保存。

如果在您的编辑会话中调整您的策略结构，则 IAM 根据以下情况确定是否保存结构调整：

使用此编辑器选项	如果编辑您的策略	然后从此选项卡中选择下一步	在选择保存更改时
可视化	已编辑	可视化	将调整策略结构
可视化	已编辑	JSON	将调整策略结构
可视化	未编辑	可视化	将调整策略结构
JSON	已编辑	可视化	将调整策略结构
JSON	已编辑	JSON	不会更改策略结构
JSON	未编辑	JSON	不会更改策略结构

IAM 可能会调整复杂策略或具有权限块或语句以允很多个服务、资源类型或条件键的策略的结构。

在可视化编辑器中选择资源 ARN

在使用可视化编辑器创建或编辑策略时，您必须先选择一个服务，然后从该服务中选择操作。如果选择的服务和操作支持选择特定的资源，可视化编辑器将列出支持的资源类型。然后，您可以选择添加 ARN 以提供有关您的资源的详细信息。您可以从以下选项中进行选择以添加资源类型的 ARN。

- 使用 ARN 生成器：根据资源类型，您可能会看到用于生成 ARN 的不同字段。您也可以选择任意，以便为指定的设置的任何值提供权限。例如，如果选择 Amazon EC2 Read（读取）访问级别组，则您的策略中的操作支持 `instance` 资源类型。为您的资源提供区域、账户和实例 ID 值。如果提供您的账户 ID，并为区域和实例 ID 选择任意，则策略为您账户中的任何实例授予权限。
- 键入或粘贴 ARN - 您可以按 Amazon Resource Name (ARN) 指定资源。您可以在 ARN 的任何字段中包含通配符 (*)（每对冒号之间）。有关更多信息，请参阅 IAM JSON 策略元素：Resource。

在可视化编辑器中拒绝权限

默认情况下，使用可视化编辑器创建的策略允许执行选择的操作。要拒绝选择的操作，请选择切换到拒绝权限。由于默认拒绝请求，建议您仅允许用户所需的操作和资源的权限。只有在要覆盖其他语句或策略单独允许的权限时，您才应创建拒绝语句。我们建议您将拒绝权限数限制为最低，因为它们可能会增加解决权限问题的难度。有关 IAM 如何评估策略逻辑的更多信息，请参阅 策略评估逻辑。

> **Note**
>
> 默认情况下，仅 Amazon Web Services 账户根用户有权访问该账户中的所有资源。因此，如果未以根用户身份登录，您必须具有策略授予的权限。

在可视化编辑器中指定多个服务

在使用可视化编辑器构建策略时，您每次只能选择一个服务。这是最佳实践，因为可视化编辑器允许您从该服务的操作中进行选择。然后，您可以从该服务和选定的操作支持的资源中进行选择。这样，就可以更轻松地创建策略和进行故障排除。

您还可以使用通配符（*）手动指定多个服务。例如，键入 **`Code*`**，以便为以 `Code` 开头的所有服务（如 `CodeBuild` 和 `CodeCommit`）提供权限。不过，您必须随后键入操作和资源 ARN 以完成您的策略。此外，在保存您的策略时，可能会调整策略结构以在单独的权限块中包含每个服务。

或者，要在服务中使用 JSON 语法（如通配符），请使用 JSON 编辑器选项创建、编辑和保存策略。

在可视化编辑器中减小策略大小

在使用可视化编辑器创建策略时，IAM 将创建一个 JSON 文档以存储您的策略。您可以切换到 JSON 编辑器选项以查看该文档。如果该 JSON 文档超过策略的大小限制，可视化编辑器将显示错误消息。您将无法查看和保存该策略。要查看 IAM 的托管策略大小限制，请参阅 IAM 和 STS 字符限制。

要在可视化编辑器中减小您的策略大小，请编辑您的策略或将权限块移到另一个策略。错误消息包括策略文档中包含的字符数。您可以使用此信息来帮助缩减策略大小。

在可视化编辑器中修复无法识别的服务、操作或资源类型

您可能会在可视化编辑器中看到一条警告，指出您的策略包含无法识别的服务、操作或资源类型。

> **Note**
>
> IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过策略模拟器测试您的策略。

如果您的策略包含无法识别的服务、操作或资源类型，则存在以下错误之一：

- 预览服务 - 处于预览状态的服务不支持可视化编辑器。如果您参与预览，则必须手动键入操作和资源 ARN 以完成您的策略。您可以忽略任何警告并继续。或者，您也可以选择 JSON 编辑器选项以键入或粘贴 JSON 策略文档。
- 自定义服务 - 自定义服务不支持可视化编辑器。如果您使用自定义服务，则必须手动键入操作和资源 ARN 以完成您的策略。您可以忽略任何警告并继续。或者，您也可以选择 JSON 编辑器选项以键入或粘贴 JSON 策略文档。
- 服务不支持可视化编辑器：如果您的策略包含不支持可视化编辑器的已正式发布（GA）的服务，则必须手动键入操作和资源 ARN 以完成您的策略。您可以忽略任何警告并继续。或者，您也可以选择 JSON 编辑器选项以键入或粘贴 JSON 策略文档。

  公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持可视化编辑器。要了解如何请求 GA 服务的可视化编辑器或策略摘要支持，请参阅服务不支持 IAM policy 摘要。
- 操作不支持可视化编辑器：如果您的策略包含的受支持服务具有不支持的操作，则必须手动键入操作和资源 ARN 以完成您的策略。您可以忽略任何警告并继续。或者，您也可以选择 JSON 编辑器选项以键入或粘贴 JSON 策略文档。

  如果您的策略包含的受支持服务具有不支持的操作，则该服务不完全支持可视化编辑器。要了解如何请求 GA 服务的可视化编辑器或策略摘要支持，请参阅服务不支持 IAM policy 摘要。
- 资源类型不支持可视化编辑器 - 如果您的策略包含的受支持操作具有不支持的资源类型，您可以忽略该警告并继续。不过，IAM 无法确认是否包含所有选定操作的资源，并且您可能会看到额外的警告。
- 拼写错误 - 在可视化编辑器中手动键入服务、操作或资源时，您创建的策略可能会包含拼写错误。建议您使用可视化编辑器，从服务和操作列表中进行选择。然后，根据提示完成资源部分。如果服务不完全支持可视化编辑器，您可能需要手动键入某些策略部分。

  如果您确定自己的策略不包含上述任何错误，那么您的策略可能包含拼写错误。检查有无以下问题：

  - 服务、操作和资源类型名称拼写错误，例如 `s2` 而不是 `s3` 或 `ListMyBuckets` 而不是 `ListAllMyBuckets`

- ARN 中不必要的文本，例如 `arn:aws:s3: : :*`
- 操作中缺少冒号，例如 `iam.CreateUser`

您可以选择下一步以查看策略摘要并确认策略是否提供所需的权限，从而对可能包含拼写错误的策略进行评估。然后，确认策略是否提供所需的权限。

排查策略摘要问题

您可以诊断并解决与策略摘要相关的问题。

缺少策略摘要

IAM 控制台中提供了策略摘要表，这些表总结了策略中对每个服务允许或拒绝的访问级别、资源和条件。策略在三个表中概括：策略摘要、服务摘要和操作摘要。策略摘要表包括由所选策略定义的服务列表和权限摘要。您可以在策略的策略详细信息页面查看附加到实体的任何策略的策略摘要。可以在 Policies 页面上查看托管策略的策略摘要。如果 Amazon 无法呈现策略摘要，您将看到 JSON 策略文档和以下错误：

无法为此策略生成摘要。您仍然可以查看或编辑 JSON 策略文档。

如果您的策略不包含摘要，则已发生下列错误之一：

- 不支持的策略元素 - IAM 不支持为包含以下策略元素之一的策略生成策略摘要：
  - `Principal`
  - `NotPrincipal`
  - `NotResource`
- 无策略权限 - 如果策略不提供任何有效权限，则无法生成策略摘要。例如，如果策略包含元素 `"NotAction": "*"` 的单个语句，则它将授予对除“所有操作”(*) 之外的所有操作的访问权限。这意味着，它未授予对任何内容的 `Deny` 或 `Allow` 访问权限。

> **Note**
>
> 在使用这些策略元素（如 `NotPrincipal`、`NotAction` 和 `NotResource`）时应谨慎。有关使用策略元素的信息，请参阅IAM JSON 策略元素参考。

如果您提供不匹配的服务和资源，则可能会创建一个不提供有效权限的策略。您指定一个服务中的操作和另一个服务中的资源时，可能会发生这种情况。在这种情况下，仍然会出现策略摘要。表示有问

题的唯一迹象是，摘要中的资源列可能包含来自不同服务的资源。如果此列包含不匹配的资源，那么您应该查看策略中是否有错误。使用策略模拟器测试您的策略，以更好地了解该策略。

摘要策略包含无法识别的服务、操作或资源类型

在 IAM 控制台中，如果策略摘要包含警告符号（ ），则策略可能包含无法识别的服务、操作或资源类型。要了解策略摘要内的警告，请参阅策略摘要（服务列表）。

> **Note**
>
> IAM 将检查支持策略摘要的服务的服务名称、操作和资源类型。但是，您的策略摘要可能包含不存在的资源值或条件。始终通过策略模拟器测试您的策略。

如果您的策略包含无法识别的服务、操作或资源类型，则存在以下错误之一：

- 预览服务 - 处于预览状态的服务不支持策略摘要。
- 自定义服务 - 自定义服务不支持策略摘要。
- 服务不支持摘要 - 如果您的策略包括不支持策略摘要的公开发布 (GA) 服务，则该服务将包含在策略摘要表的 Unrecognized services（无法识别的服务）部分。公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持 IAM policy 摘要。要了解如何请求 GA 服务的策略摘要支持，请参阅服务不支持 IAM policy 摘要。
- 操作不支持摘要 - 如果您的策略包含的受支持服务具有不支持的操作，则该操作将包含在服务摘要表的 Unrecognized actions（无法识别的操作）部分。要了解服务摘要内的警告，请参阅服务摘要（操作列表）。
- 资源类型不支持摘要 - 如果您的策略包含的受支持操作具有不支持的资源类型，则该资源将包含在服务摘要表的 Unrecognized resource types（无法识别的资源类型）部分。要了解服务摘要内的警告，请参阅服务摘要（操作列表）。
- 错字 — Amazon 检查 JSON 语法是否正确，并且策略不在策略验证中包含错字或其他错误。

> **Note**
>
> 作为最佳实践，我们建议您使用 IAM Access Analyzer 验证您的 IAM policy，以确保权限的安全性和功能性。我们建议您打开现有策略，查看并解决任何策略验证建议。

服务不支持 IAM policy 摘要

IAM 策略摘要或可视化编辑器可能不支持已正式发布（GA）的服务或操作。公开提供服务是公开发布的服务，不是预览或自定义服务。如果无法识别的服务是公开提供的，并且名称拼写正确，则该服务不支持这些功能。如果您的策略包含的受支持服务具有不受支持的操作，则表示该服务不完全支持 IAM policy 摘要。

请求服务添加 IAM policy 摘要或可视化编辑器支持

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 找到包含不支持的服务的策略：
   - 如果该策略是托管策略，请在导航窗格中选择 Policies。在策略列表中，选择要查看的策略的名称。
   - 如果该策略是附加到用户的内联策略，请在导航窗格中选择 Users。在用户列表中，选择要查看其策略的用户的名称。在用户的策略表中，展开您要查看的策略摘要的标题。
3. 在 Amazon Web Services 管理控制台页脚左侧，选择 Feedback (反馈)。在 IAM 反馈框中，键入 **`I request that the <ServiceName> service add support for IAM policy summaries and the visual editor`**。如果希望多个服务支持摘要，请键入 **`I request that the <ServiceName1>, <ServiceName2>, and <ServiceName3> services add support for IAM policy summaries and the visual editor`**。

请求服务为缺失的操作增加 IAM policy 摘要支持

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 找到包含不支持的服务的策略：
   - 如果该策略是托管策略，请在导航窗格中选择 Policies。在策略列表中，选择要查看的策略的名称。

- 如果该策略是附加到用户的内联策略，请在导航窗格中选择 Users。在用户列表中，选择要查看其策略的用户的名称。在用户的策略表中，选择要查看的策略的名称以展开策略摘要。

3. 在策略摘要中，选择包含不支持操作的服务的名称。
4. 在 Amazon Web Services 管理控制台页脚左侧，选择 Feedback (反馈)。在 IAM 反馈框中，键入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName> action**。如果要报告多个不支持的操作，请键入 **I request that the <ServiceName> service add IAM policy summary and the visual editor support for the <ActionName1>, <ActionName2>, and <ActionName3> actions**。

要请求包含缺少操作的不同服务，请重复最后三个步骤。

我的策略未授予预期权限

要给用户、组、角色或资源指定权限，您必须创建一个策略，它是一个定义权限的文档。策略文档包含以下元素：

- 效果 - 策略允许还是拒绝访问
- 操作 - 策略允许或拒绝的操作的列表
- 资源 - 作为操作目标的资源的列表
- 条件（可选）- 策略在哪些情况下授予权限

要了解上述及其他策略元素，请参阅IAM JSON 策略元素参考。

要授予访问权限，策略必须定义具有支持资源的操作。如果策略还包含条件，该条件必须包含全局条件键或必须应用至操作。要了解操作支持哪些资源，请参阅服务的 Amazon 文档。要了解操作支持哪些条件，请参阅 Amazon 服务的操作、资源或条件键。

请检查策略是否定义了未授予权限的操作、资源或条件。使用 IAM 控制台查看您的策略的策略摘要，网址为 https://console.aws.amazon.com/iam/。您可以使用策略摘要识别和纠正策略中的问题。

对于元素不按 IAM policy 中的定义授予权限的情况，原因可能有：

- 定义了操作，但没有适用的资源
- 定义了资源，但没有适用的操作
- 定义了条件，但没有适用的操作

要查看包含警告的策略摘要的示例，请参阅the section called “策略摘要（服务列表）”。

定义了操作，但没有适用的资源

下面的策略定义了所有 ec2:Describe* 操作和一个特定资源。这些操作都不支持资源级权限，因而未授予任何 ec2:Describe 操作。资源级权限意味着操作支持使用策略的 Resource 元素中的 ARN 的资源。如果操作不支持资源级权限，则策略中的语句必须在 * 元素中使用通配符 (Resource)。要了解哪些服务支持资源级权限，请参阅使用 IAM 的 Amazon 服务。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:Describe*",
            "Resource": "arn:aws:ec2:us-east-2:111122223333:instance/*"
        }
    ]
}
```

此策略未提供任何权限，并且策略摘要包含以下错误：

```
This policy does not grant any permissions. To grant access, policies must
have an action that has an applicable resource or condition.
```

要修复该策略，必须在 * 元素中使用 Resource。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "ec2:Describe*",
        "Resource": "*"
    }]
}
```

定义了资源，但没有适用的操作

下面的策略定义了 Amazon S3 存储桶资源，但不包含可在该资源上执行的 S3 操作。该策略还向所有 Amazon CloudFront 操作授予完全访问权限。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
        "Effect": "Allow",
        "Action": "cloudfront:*",
        "Resource": [
            "arn:aws:cloudfront:*",
            "arn:aws:s3:::amzn-s3-demo-bucket"
        ]
        }
    ]
}
```

该策略为所有 CloudFront 操作提供权限。但由于策略定义了 S3 amzn-s3-demo-bucket 资源而未定义任何 S3 操作，策略摘要将包含以下警告：

```
This policy defines some actions, resources, or conditions that do not
provide permissions. To grant access, policies must have an action that has
an applicable resource or condition.
```

要修复该策略以提供 S3 存储桶权限，必须定义可在存储桶资源上执行的 S3 操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudfront:*",
                "s3:CreateBucket",
```

```
                "s3:ListBucket*",
                "s3:PutBucket*",
                "s3:GetBucket*"
            ],
            "Resource": [
                "arn:aws:cloudfront:*",
                "arn:aws:s3:::amzn-s3-demo-bucket"
            ]
        }
    ]
}
```

此外，要修复该策略以只提供 CloudFront 权限，应删除 S3 资源。

定义了条件，但没有适用的操作

如果 S3 前缀等于 `custom` 且版本 ID 等于 `1234`，下面的策略为所有 S3 资源定义两个 Amazon S3 操作。但是，`s3:VersionId` 条件键用于对象版本标记，不受所定义的存储桶操作的支持。要了解操作支持哪些条件，请参阅 Amazon 服务的操作、资源和条件键，然后选择服务以查看条件键的服务文档。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
```

```
                }
            }
        }
    ]
}
```

如果存储桶名称包含 s3:ListBucketVersions 前缀，该策略为 s3:ListBucket 操作和 custom 操作提供权限。但是，由于任何定义的操作都不支持 s3:VersionId 条件，策略摘要将包含以下错误：

```
This policy does not grant any permissions. To grant access, policies must have an action that has an applicable resource or condition.
```

要修复该策略以使用 S3 对象版本标记，就必须定义支持 s3:VersionId 条件键的 S3 操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket",
                "s3:GetObjectVersion"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": [
                        "custom"
                    ],
                    "s3:VersionId": [
                        "1234"
                    ]
                }
            }
        }
    ]
}
```

此策略为策略中的每个操作和条件提供权限。但由于不存在单一操作匹配两个条件的情况，策略不提供任何权限。为解决这一问题，您必须创建两个单独的语句，每个语句只包含具有要应用条件的操作。

要修复此策略，请创建两个语句。第一个语句包含支持 s3:prefix 条件的操作，第二个语句包含支持 s3:VersionId 条件的操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:ListBucketVersions",
                "s3:ListBucket"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:prefix": "custom"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "s3:GetObjectVersion",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "s3:VersionId": "1234"
                }
            }
        }
    ]
}
```

策略管理故障排查

您可以诊断并解决与策略管理相关的问题。

在 IAM 账户中附加或分离策略

有些 Amazon 托管策略与服务关联。这些策略仅用于该服务的某个服务相关角色。在 IAM 控制台中查看策略详细信息页面时，该页面包含一个横幅，指示该策略与服务关联。您不能将该策略附加到 IAM 内的用户、组或角色。当您为服务创建服务相关角色时，该策略会自动附加到您的新角色。由于策略是必需的，因此您无法将策略与服务相关角色分离。

根据您的 IAM 身份活动更改这些身份的策略

您可以根据 IAM 身份（用户、组和角色）的活动更新其策略。要执行此操作，请在 CloudTrail 事件历史记录中查看您账户的事件。CloudTrail 事件日志包含详细的事件信息，您可以用来更改策略的权限。

用户或角色尝试在 Amazon 中执行操作，该请求被拒绝。

> 考虑用户或角色是否应具有执行该操作的权限。如果应该具有，您可以添加操作，甚至添加它们试图根据其策略访问的资源的 ARN。

用户或角色具有其不使用的权限。

> 考虑从其策略中删除这些权限。请确保您的策略仅授予执行必需的操作所需要的最低权限。

有关使用 CloudTrail 的更多信息，请参阅 Amazon CloudTrail 用户指南中的在 CloudTrail 控制台中查看 CloudTrail 事件。

JSON 策略文档故障排查

您可以诊断并解决与 JSON 策略文档相关的问题。

验证您的策略

当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可操作建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

我在 JSON 编辑器中没有用于策略验证的权限

在 Amazon Web Services 管理控制台 中，如果您没有查看 IAM Access Analyzer 策略验证结果的权限，则可能会收到以下错误：

```
You need permissions. You do not have the permissions required to perform
this operation. Ask your administrator to add permissions.
```

要纠正该错误，请要求管理员为您添加 access-analyzer:ValidatePolicy 权限。

多个 JSON 策略对象

IAM 策略必须仅包含一个 JSON 对象。可通过在两旁放置 { } 括号来表示对象。您可以通过在外部对中嵌入额外的 {} 大括号，将其他对象嵌套在 JSON 对象中。策略必须仅包含一对最外层的 {} 大括号。以下示例不正确，因为它在顶层包含两个对象 (以##标示)：

JSON

```
{
      "Version":"2012-10-17",
      "Statement":
      {
         "Effect":"Allow",
         "Action":"ec2:Describe*",
         "Resource":"*"
      }
    }
    {
      "Statement": {
         "Effect": "Allow",
         "Action": "s3:*",
         "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    }
```

不过，您可以使用正确的策略语法来实现上面示例的意图。可以将两个数据块合并到单个 Statement 元素中，而不是包含两个完整的策略对象（每个都有自己的 Statement 元素）。Statement 元素将两个对象组成的数组作为其值，如以下示例所示 (以粗体标示)：

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
```

```
      },
      {
        "Effect": "Allow",
        "Action": "s3:*",
        "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
      }
    ]
  }
```

多个 JSON Statement 元素

此错误乍一看似乎是由上一部分变化而来的。但是，它在语法上是不同类型的错误。在以下示例中，顶层只有一个策略对象，由单个 { } 括号对表示。但是，该对象包含两个 Statement 元素。

一个 IAM policy 只能包含一个 Statement 元素，名称 (Statement) 在冒号左侧，它的值在冒号右侧。Statement 元素的值必须是对象，以 { } 括号表示，其中包含一个 Effect 元素、一个 Action 元素和一个 Resource 元素。以下示例不正确，因为策略对象包含两个 Statement 元素 (以##标示)：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "ec2:Describe*",
      "Resource": "*"
    },
    "Statement": {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  }
```

值对象可以是多个值对象组成的数组。为解决此问题，可使用对象数组将两个 Statement 元素组合为一个元素，如下例所示 (以粗体标示)：

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [
        {
          "Effect": "Allow",
          "Action": "ec2:Describe*",
          "Resource":"*"
        },
        {
          "Effect": "Allow",
          "Action": "s3:*",
          "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
        }
     ]
    }
```

Statement 元素的值是对象数组。此示例中的数组包含两个对象，每个对象自身是 Statement 元素的正确值。数组中的每个对象之间用逗号隔开。

在 JSON Statement 元素中具有多个 Effect、Action 或 Resource 元素

在 Statement 名称/值对的值侧，对象只能包含一个 Effect 元素、一个 Action 元素和一个 Resource 元素。以下策略不正确，因为其在 Statement 中有两个 Effect 元素：

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Effect": "Allow",
        "Action": "ec2:* ",
        "Resource": "*"
      }
    }
```

> **Note**
>
> 策略引擎不允许新的或已编辑的策略中出现此类错误。但是，策略引擎仍然允许在引擎更新之前保存的策略。现有策略的错误行为如下所示：
>
> - 多个 `Effect` 元素：仅遵循最后一个 `Effect` 元素。忽略其他元素。
> - 多个 `Action` 元素：所有 `Action` 元素进行内部合并，被视为单个列表。
> - 多个 `Resource` 元素：所有 `Resource` 元素进行内部合并，被视为单个列表。
>
> 策略引擎不允许您保存任何有 语法 错误的策略。请先更正策略中的错误，然后再保存。查看并更正针对您的策略的任何策略验证建议。

在每种情况下，解决方案都是删除不正确的多余元素。对于 `Effect` 元素，这十分简单：如果您希望前面的示例拒绝 针对 Amazon EC2 实例的权限，则必须从策略中删除行 `"Effect": "Allow",`，如下所示：

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
        "Effect": "Deny",
        "Action": "ec2:*",
        "Resource": "*"
      }
    }
```

但是，如果重复元素是 `Action` 或 `Resource`，则解决方法可能会更加复杂。您可能需要向多个操作允许 (或拒绝) 权限，或者需要控制对多个资源的访问。例如，以下示例不正确，因为它有多个 `Resource` 元素 (以##标示)：

JSON

```
{
      "Version":"2012-10-17",
      "Statement": {
```

```
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
    }
  }
```

Statement 元素的值对象中的每个必需元素都只能出现一次。解决方案是将每个值置于数组中。以下示例通过将两个单独的 Resource 元素合并为一个以数组为值对象的 Resource 元素来对此进行说明 (以粗体标示) :

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ]
    }
  }
```

缺少 JSON Version 元素

Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。相比之下，当您在 IAM 中更改客户管理型策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅IAM JSON 策略元素：Version。要了解策略版本的更多信息，请参阅the section called “IAM 策略版本控制”。

随着 Amazon 功能的发展，为支持这些功能，IAM policy 增加了新的功能。有时，策略语法更新包括新版本号。如果您在策略中使用策略语法的较新功能，则必须向策略分析引擎告知所使用的版本。默认策略版本是“2008-10-17”。如果要使用之后引入的任何策略功能，则必须指定支持所需功能的版本号。我们建议始终 包含最新的策略语法版本号 (目前为 "Version": "2012-10-17")。例如，以下策略是不正确的，因为它在资源的 ARN 中使用 ${...} 策略变量。但是，它无法指定支持该策略变量的策略语法版本（以## 标示）：

```
{
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

通过在策略顶层添加值为 2012-10-17（支持策略变量的第一个 IAM API 版本）的 Version 元素，可解决此问题（以粗体标示）：

JSON

```
{
  "Version":"2012-10-17",
  "Statement":
  {
    "Action": "iam:*AccessKey*",
    "Effect": "Allow",
    "Resource": "arn:aws:iam::123456789012:user/${aws:username}"
  }
}
```

排查通行密钥和 FIDO 安全密钥问题

使用此处的信息可帮助您诊断和修复在使用 FIDO2 安全密钥时可能遇到的常见问题。

主题

- 我无法启用我的 FIDO 安全密钥
- 我无法使用 FIDO 安全密钥登录
- 我的 FIDO 安全密钥已丢失或损坏
- 其它问题

我无法启用我的 FIDO 安全密钥

请参阅以下解决方案，具体取决于您的身份是 IAM 用户还是系统管理员

IAM 用户

如果您无法启用 FIDO 安全密钥，请检查以下各项：

- 您使用的配置是否受支持？

  IAM 支持 FIDO2 安全设备，这些设备可通过 USB、Bluetooth或 NFC 连接到您的设备。IAM 还支持 TouchID 或 FaceID 等平台身份验证器。IAM 不支持 Windows Hello 的本地密钥注册。若要创建和使用密钥，Windows 用户应使用跨设备身份验证，即使用来自一台设备（例如移动设备）的密钥或硬件安全密钥在另一台设备（如笔记本电脑）上登录。

  有关可与 WebAuthn 和 Amazon 结合使用的设备和浏览器的信息，请参阅 使用密钥或安全密钥的受支持配置。
- 您是否使用了任何浏览器插件？

  Amazon 不支持使用插件来添加 WebAuthn 浏览器支持。请改用可对 WebAuthn 标准提供本机支持的浏览器。

  即使您使用的是支持的浏览器，也可能有插件与 WebAuthn 不兼容。不兼容的插件可能会使您无法启用和使用符合 FIDO 的安全密钥。禁用任何可能不兼容的插件，然后重新启动浏览器。然后，重新尝试启用 FIDO 安全密钥。
- 您是否具有适当的权限？

  如果没有上述任何兼容性问题，可能您不具有适当的权限。请联系您的系统管理员。

系统管理员

尽管您的 IAM 用户使用了支持的配置，如果仍无法启用其 FIDO 安全密钥，则请检查其权限。有关详细示例，请参阅 IAM 教程：允许用户管理其凭证和 MFA 设置。

我无法使用 FIDO 安全密钥登录

如果您无法使用 FIDO 安全密钥登录 Amazon Web Services 管理控制台，则请先参阅 使用密钥或安全密钥的受支持配置。如果您使用的是受支持的配置，但无法登录，请联系您的系统管理员以获得帮助。

我的 FIDO 安全密钥已丢失或损坏

最多可以向用户分配 8 台当前支持的 MFA 类型的任意组合的 MFA 设备。分配多台 MFA 设备时，您只需要一个 MFA 设备即可登录 Amazon Web Services 管理控制台。更换 FIDO 安全密钥与更换硬件

TOTP 令牌相似。如果丢失或损坏任何类型的 MFA 设备，则请参阅 在 IAM 中恢复受 MFA 保护的身份。

其它问题

如果您遇到此处未涉及的 FIDO 安全密钥问题，请执行下列操作之一：

- IAM 用户：请联系您的系统管理员。
- Amazon Web Services 账户 根用户：请联系 Amazon Support。

排查 IAM 角色问题

使用此处的信息可帮助您诊断和修复在使用 IAM 角色时可能遇到的常见问题。

主题

- 我无法代入角色
- 我的 Amazon 账户中出现新角色
- 我无法编辑或删除我的 Amazon Web Services 账户 中的角色
- 我无权执行：iam:PassRole
- 为什么我无法在 12 小时会话中担任角色？ (Amazon CLI、Amazon API)
- 当我尝试在 IAM 控制台中切换角色时收到错误
- 我的角色具有一个允许我执行操作的策略，但出现“访问被拒绝”错误
- 服务未创建角色的默认策略版本
- 控制台中没有服务角色的使用案例

我无法代入角色

请检查以下事项：

- 要允许用户在角色会话中重新代入当前角色，请将角色 ARN 或 Amazon Web Services 账户 ARN 指定为角色信任策略中的主体。提供计算资源（例如 Amazon EC2、Amazon ECS、Amazon EKS 和 Lambda）的 Amazon Web Services 服务会提供临时凭证并自动更新这些凭证。这样可以确保您始终拥有一组有效的凭证。对于这些服务，无需重新代入当前角色即可获得临时凭证。但是，如果您需要传递 会话标签 或者 会话策略，则需要重新代入当前角色。要了解如何修改角色信任策略以添加主体角色 ARN 或 Amazon Web Services 账户 ARN，请参阅 更新角色信任策略。

- 当您使用 Amazon Web Services 管理控制台 担任角色时，确保使用角色的确切名称。担任角色时，角色名称区分大小写。
- 使用 Amazon STS API 或 Amazon CLI 担任角色时，请确保使用 ARN 中角色的确切名称。担任角色时，角色名称区分大小写。
- 当您使用基于 SAML 的联合身份验证身份提供商代入角色，并且启用了 SAML 加密时，请确保已为 SAML 身份提供商上传了有效的私有解密密钥。有关更多信息，请参阅 管理 SAML 加密密钥。
- 验证您的 IAM policy 是否为您授予为要担任的角色调用 `sts:AssumeRole` 的权限。您的 IAM policy 的 `Action` 元素必须允许您调用 `AssumeRole` 操作。此外，您的 IAM policy 的 `Resource` 元素必须指定您要带入的角色。例如，`Resource` 元素可以按其 Amazon Resource Name (ARN) 或按通配符 (*) 指定角色。例如，至少一个适用于您的策略必须授予与以下权限类似的权限：

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
```

- 验证您的 IAM 身份是否标记有 IAM policy 需要的任何标签。例如，在以下策略权限中，`Condition` 元素要求您（因为主体请求担任该角色）必须具有特定标签。您必须被标记有 `department = HR` 或 `department = CS`。否则，您无法担任该角色。要了解如何标记 IAM 用户和角色，请参阅 the section called “IAM 资源的标签”。

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "*",
"Condition": {"StringEquals": {"aws:PrincipalTag/department": [
        "HR",
        "CS"
    ]}}
```

- 确保您满足角色的信任策略中指定的所有条件。`Condition` 可以指定有效期、外部 ID 或请求必须来自于特定 IP 地址。请考虑以下示例：如果当前日期是指定日期以后的任意时间，则策略根本不会匹配，并且无法为您授予担任该角色的权限。

```
"Effect": "Allow",
"Action": "sts:AssumeRole",
"Resource": "arn:aws:iam::account_id_number:role/role-name-you-want-to-assume"
"Condition": {
    "DateLessThan" : {
        "aws:CurrentTime" : "2016-05-01T12:00:00Z"
    }
```

```
    }
```

- 验证您用于调用 AssumeRole 的 Amazon Web Services 账户 是否为您要代入的角色的受信任实体。在角色的信任策略中将受信任实体定义为 Principal。以下示例是一个信任策略，它附加到您要担任的角色。在该示例中，您登录时使用的 IAM 用户的账户 ID 必须是 123456789012。如果您的账号未在角色信任策略的 Principal 元素中列出，则无法担任该角色。在访问策略中为您授予何种权限无关紧要。注意，示例策略将权限限制为在 2017 年 7 月 1 日至 2017 年 12 月 31 日 (UTC 时间，这两个日期也包含在内) 之间发生的操作。如果您在上述日期范围之前或之后登录，则策略不匹配，您无法担任该角色。

```
    "Effect": "Allow",
    "Principal": { "AWS": "arn:aws:iam::123456789012:root" },
    "Action": "sts:AssumeRole",
    "Condition": {
      "DateGreaterThan": {"aws:CurrentTime": "2017-07-01T00:00:00Z"},
      "DateLessThan": {"aws:CurrentTime": "2017-12-31T23:59:59Z"}
    }
```

- 来源身份 — 管理员可以配置角色以要求身份来传递自定义字符串，该字符串标识在 Amazon 中执行操作的个人或应用程序，称为源身份。验证所担任的角色是否需要设置源身份。更多有关源身份的信息，请参阅 监控和控制使用所担任角色执行的操作。

我的 Amazon 账户中出现新角色

某些 Amazon 服务要求您使用直接与该服务相链接的唯一服务角色类型。该服务相关角色由服务预定义，具有服务所需的所有权限。这样您就不必手动添加必要的权限，从而能够更轻松地设置服务。有关服务相关角色的一般信息，请参阅创建服务相关角色。

在开始支持服务相关角色时，您可能已在使用服务。如果事实如此，您可能会收到一封电子邮件，告知您账户中将有新角色。该角色包括此项服务代表您执行操作所需的所有权限。您不需要执行任何操作支持该角色。但是，您不能从账户中删除该角色。这样会删除此项服务访问 Amazon 资源所需的权限。您可以转至 IAM 控制台的 IAM Roles（角色）页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities（可信实体）列中随 (Service-linked role) [（服务相关角色）] 一起显示。

有关哪些服务支持服务相关角色的信息，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色列为是的服务。有关使用某个服务的服务相关角色的信息，请选择 Yes 链接。

我无法编辑或删除我的 Amazon Web Services 账户 中的角色

您不能删除或编辑 IAM 中服务相关角色的权限。这些角色包括服务代表您执行操作所需的预定义信任和权限。您可以使用 IAM 控制台、Amazon CLI 或 API 仅编辑服务相关角色的描述。您可以转至控制台中的 IAM Roles（角色）页面来查看您的账户中的服务相关角色。服务相关角色在表的 Trusted entities（可信实体）列中随 (Service-linked role) [（服务相关角色）] 一起显示。角色的 Summary 页面上的横幅也指示角色是服务相关角色。您只能通过链接的服务管理和删除这些角色 (如果该服务支持此操作)。修改或删除服务相关角色时要小心，因为这样做可能会删除服务访问 Amazon 资源所需的权限。

有关哪些服务支持服务相关角色的信息，请参阅使用 IAM 的 Amazon 服务并查找服务相关角色列为是的服务。

我无权执行：iam:PassRole

在创建服务相关角色时，您必须有权将该角色传递给服务。在某些服务中执行操作时，该服务自动在您的账户中创建一个服务相关角色。例如，在首次创建 Auto Scaling 组时，Amazon EC2 Auto Scaling 为您创建 AWSServiceRoleForAutoScaling 服务相关角色。如果您尝试创建 Auto Scaling 组而没有 PassRole 权限，则会出现以下错误：

```
ClientError: An error occurred (AccessDenied) when calling the
PutLifecycleHook operation: User: arn:aws:sts::111122223333:assumed-role/
Testrole/Diego is not authorized to perform: iam:PassRole on resource:
arn:aws:iam::111122223333:role/aws-service-role/autoscaling.amazonaws.com/
AWSServiceRoleForAutoScaling
```

要纠正该错误，请要求管理员为您添加 iam:PassRole 权限。

要了解哪些服务支持服务相关角色，请参阅使用 IAM 的 Amazon 服务。要了解服务是否自动为您创建服务相关角色，请选择是链接以查看服务的服务相关角色文档。

为什么我无法在 12 小时会话中担任角色？ (Amazon CLI、Amazon API)

在使用 Amazon STS AssumeRole* API 或 assume-role* CLI 操作担任角色时，您可以为 DurationSeconds 参数指定一个值。您可以指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的值。如果指定的值高于该设置，操作将失败。该设置的最大值为 12 小时。例如，如果您指定的会话持续时间为 12 小时，但管理员设置的最大会话持续时间为 6 小时，您的操作将失败。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。

如果您使用角色链 (使用一个角色担任另一个角色)，您的会话限制为最多 1 小时。如果您随后使用 DurationSeconds 参数提供大于 1 小时的值，操作将失败。

当我尝试在 IAM 控制台中切换角色时收到错误

您在切换角色页上输入的信息必须与角色的信息匹配。否则，操作失败，您会收到以下错误：

```
Invalid information in one or more fields. Check your information or
contact your administrator.
```

如果您收到此错误，请确认以下信息是否正确：

- 账户 ID 或别名 – Amazon Web Services 账户 ID 是一组 12 位的数字。您的账户可能有一个别名，这是一个易记标识符，例如您的公司名称，可用来代替您的 Amazon Web Services 账户 ID。您可以在此字段中使用账户 ID 或别名。
- 角色名称 - 角色名称区分大小写。账户 ID 和角色名称必须与为角色配置的内容匹配。

如果您继续收到错误消息，请与您的管理员联系以验证以前的信息。角色信任策略或 IAM 用户策略可能会限制您的访问权限。您的管理员可以验证这些策略的权限。

我的角色具有一个允许我执行操作的策略，但出现“访问被拒绝”错误

您的角色会话可能受会话策略的限制。以编程方式使用 Amazon STS 请求临时安全凭证时，您可以选择传递内联或托管会话策略。会话策略是高级策略，在以编程方式为角色创建临时凭证会话时，这些策略将作为参数进行传递。您可以使用 Policy 参数传递单个 JSON 内联会话策略文档。您可以使用 PolicyArns 参数指定最多 10 个托管会话策略。生成的会话的权限是角色的基于身份的策略与会话策略的交集。或者，如果您的管理员或自定义程序为您提供临时凭证，它们可能已包含会话策略以限制您的访问。

服务未创建角色的默认策略版本

服务角色是服务代表您在您的账户中执行操作而担任的角色。在设置一些 Amazon 服务环境时，您必须为服务定义要代入的角色。在某些情况下，服务会在 IAM 中为您创建服务角色及其策略。尽管您可以从 IAM 内部修改或删除服务角色及其策略，但 Amazon 不建议这样做。角色和策略仅供该服务使用。如果编辑策略并设置另一个环境，则当服务尝试使用相同的角色和策略时，操作可能会失败。

例如，首次使用 Amazon CodeBuild 时，服务会创建一个名为 codebuild-RWBCore-service-role 的角色。该服务角色使用名为 codebuild-RWBCore-managed-policy 的策略。如果编辑

策略，它会创建一个新版本并将该版本保存为默认版本。如果您在 Amazon CodeBuild 中执行后续操作，服务可能会尝试更新策略。如果是这样，您会收到以下错误：

```
codebuild.amazon.com did not create the default version (V2) of the
codebuild-RWBCore-managed-policy policy that is attached to the codebuild-
RWBCore-service-role role. To continue, detach the policy from any other
identities and then delete the policy and the role.
```

如果您收到此错误，则必须在 IAM 中进行更改，然后才能继续服务操作。首先，将默认策略版本设置为 V1，然后重试该操作。如果先前删除了 V1，或者如果选择 V1 不起作用，则清理并删除现有策略和角色。

有关编辑托管策略的更多信息，请参阅编辑客户托管策略（控制台）。有关策略版本的更多信息，请参阅IAM 策略版本控制。

删除服务角色及其策略

1. 登录到 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略。
3. 在策略列表中，选择要删除的策略的名称。
4. 选择附加的实体选项卡可查看哪些 IAM 用户、组或角色使用此策略。如果这些身份中的任何一个使用此策略，请完成以下任务：
   a. 创建具有必要权限的新托管策略。要确保这些身份在操作之前和之后具有相同的权限，请从现有策略中复制 JSON 策略文档。然后创建新的管理型策略并粘贴 JSON 文档，如使用 JSON 编辑器创建策略中所述。
   b. 对于每个受影响的身份，附加新策略，然后分离旧策略。有关更多信息，请参阅 添加和删除 IAM 身份权限。
5. 在导航窗格中，选择角色。
6. 在角色列表中，选择要删除的角色的名称。
7. 选择信任关系选项卡以查看哪些实体可以代入该角色。如果列出了服务以外的任何实体，请完成以下任务：
   a. 创建信任这些实体的新角色。
   b. 您在上一步中创建的策略。如果跳过了该步骤，请立即创建新的托管策略。

c. 通知代入该角色的任何人，他们不能再这样做。向他们提供有关如何代入新角色并具有相同权限的信息。

8. 删除策略。
9. 删除角色。

控制台中没有服务角色的使用案例

某些服务要求您手动创建服务角色才能向服务授予权限，以代表您执行操作。如果该服务未在 IAM 控制台中列出，您必须手动将该服务列为受信任的主体。如果您正在使用的服务或功能的文档中不包含将服务列为受信任主体的说明，请为该页面提供反馈。

要手动创建服务角色，您必须知道将担任该角色的服务的服务主体。服务主体是一个标识符，用于向服务授予权限。服务主体由服务定义。

您可以通过检查以下内容来查找某些服务的服务主体：

1. 打开 使用 IAM 的 Amazon 服务。
2. 检查 Service-linked roles（服务相关角色）列中是否具有 Yes（是）的服务。
3. 选择是链接以查看该服务的服务相关角色文档。
4. 查找该服务的“服务相关角色权限”部分，查看服务主体。

您可以使用 Amazon CLI 命令或 Amazon API 操作手动创建服务角色。要使用 IAM 控制台手动创建服务角色，请完成以下任务：

1. 使用您的账户 ID 创建 IAM 角色。请勿附加策略或授予任何权限。有关更多信息，请参阅 创建角色，向 IAM 用户授予权限。
2. 打开角色并编辑信任关系。角色必须信任该服务，而不是信任该账户。例如，更新以下 `Principal` 元素：

   ```
   "Principal": { "AWS": "arn:aws:iam::123456789012:root" }
   ```

   将主体更改为服务的值，例如 IAM。

   ```
   "Principal": { "Service": "iam.amazonaws.com" }
   ```

3. 通过将权限策略附加到角色来添加服务所需的权限。

4. 返回到需要权限的服务，然后使用记录的方法将新的服务角色通知服务。

排查 IAM 和 Amazon EC2 问题

以下信息可帮助您排查与 Amazon EC2 相关的 IAM 问题。

主题

- 当尝试启动实例时，我没有在 Amazon EC2 控制台的 IAM 角色列表中看到该角色
- 我的实例的凭证适用于错误的角色
- 当时尝试调用 AddRoleToInstanceProfile 时，发生 AccessDenied 错误
- Amazon EC2：在我尝试使用角色启动实例时，出现 AccessDenied 错误
- 我无法访问我的 EC2 实例的临时安全凭证
- IAM 树形子目录中 info 文档的错误意味着什么？

当尝试启动实例时，我没有在 Amazon EC2 控制台的 IAM 角色列表中看到该角色

请检查以下事项：

- 如果您以 IAM 用户身份登录，请验证您是否有调用 `ListInstanceProfiles` 的权限。有关使用角色所需权限的信息，请参阅 在 Amazon EC2 中使用角色所需的权限。有关为用户添加权限的信息，请参阅 管理 IAM 策略。

  如果您无法修改自己的权限，您必须联系可以使用 IAM 的管理员来更新您的权限。
- 如果您使用 IAM CLI 或 API 创建角色，则请验证以下内容：
  - 您已创建实例配置文件，并将角色添加到该实例配置文件。
  - 您对角色和实例配置文件使用相同的名称。如果角色和实例配置文件的名称不同，则无法在 Amazon EC2 控制台中看到正确的角色名称。

  Amazon EC2 控制台的 IAM 角色列表列明了实例配置文件的名称，而不是角色名称。您必须选择包含所需角色的实例配置文件的名称。有关实例配置文件的详细信息，请参阅 使用实例配置文件。

> **Note**
>
> 如果您使用 IAM 控制台创建角色，则不需要使用实例配置文件。对于您在 IAM 控制台中创建的每个角色，使用与角色相同的名称创建实例配置文件，而且角色将自动添加到实例配置文件。一个实例配置文件只能包含一个 IAM 角色，不能提高该限制。

我的实例的凭证适用于错误的角色

最近可能替换了实例配置文件中的角色。如果是，您的应用程序需要等待下一次自动计划的凭证轮换，然后才能使用您的角色的凭证。

要强制进行更改，您必须取消关联实例配置文件，然后再次关联实例配置文件；您也可以停止并重新启动实例。

当时尝试调用 **AddRoleToInstanceProfile** 时，发生 **AccessDenied** 错误

如果您以 IAM 用户身份发出请求，请验证您拥有以下权限：

- `iam:AddRoleToInstanceProfile` 对匹配实例配置文件 ARN (例如 `arn:aws:iam::999999999999:instance-profile/ExampleInstanceProfile`) 的资源执行操作的权限。

有关使用角色所需权限的更多信息，请参阅 怎样入门？。有关为用户添加权限的信息，请参阅 管理 IAM 策略。

Amazon EC2：在我尝试使用角色启动实例时，出现 **AccessDenied** 错误

请检查以下事项：

- 发布不包含实例配置文件的实例。这样做可以帮助您确认问题的范围在 Amazon EC2 实例的 IAM 角色中。
- 如果您以 IAM 用户身份发出请求，请验证您拥有以下权限：
  - 对通配符资源 ("*") 执行 `ec2:RunInstances` 操作的许可
  - `iam:PassRole` 对匹配角色 ARN (例如 `arn:aws:iam::999999999999:role/ExampleRoleName`) 的资源执行操作的权限。

- 调用 IAM `GetInstanceProfile` 操作，确认您正使用有效的实例配置文件名称或有效的实例配置文件 ARN。有关更多信息，请参见通过 Amazon EC2 实例使用 IAM 角色。
- 调用 IAM `GetInstanceProfile` 操作，确认实例配置文件中包含角色。空的实例配置文件将发生 `AccessDenied` 错误，运行失败。有关创建角色的更多信息，请参阅 IAM 角色创建。

有关使用角色所需权限的更多信息，请参阅 怎样入门？。有关为用户添加权限的信息，请参阅 管理 IAM 策略。

我无法访问我的 EC2 实例的临时安全凭证

要访问 EC2 实例上的临时安全凭证，您必须首先使用 IAM 控制台创建角色。然后，您使用该角色启动 EC2 实例并检查运行中的实例。有关更多信息，请参阅 使用 IAM 角色向在 Amazon EC2 实例上运行的应用程序授予权限 中的怎样入门？。

如果您仍然无法访问 EC2 实例上的临时安全凭证，请检查以下内容：

- 您能否访问实例元数据服务 (IMDS) 的其他部分？ 如果不能，请检查确认没有阻止向 IMDS 发送请求的防火墙规则。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/
hostname; echo
```

- 是否存在 IMDS 的 `iam` 子树？ 如果不存在，请调用 EC2 `DescribeInstances` API 操作或者使用 `aws ec2 describe-instances` CLI 命令，验证您的实例是否具有与其关联的 IAM 实例配置文件。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam;
 echo
```

- 检查 IAM 子树中的 `info` 文档是否指示发生错误。如果发生错误，请参阅 IAM 树形子目录中 `info` 文档的错误意味着什么？ 来了解更多信息。

```
[ec2-user@domU-12-31-39-0A-8D-DE ~]$ GET http://169.254.169.254/latest/meta-data/iam/
info; echo
```

IAM 树形子目录中 **info** 文档的错误意味着什么？

iam/info 文档指示 **"Code":"InstanceProfileNotFound"**

您的 IAM 实例配置文件已经被删除，Amazon EC2 无法再为您的实例继续提供凭证。您必须将有效的实例配置文件附加到您的 Amazon EC2 实例。

如果存在具有该名称的实例配置文件，请检查是该实例配置文件没有被删除还是以相同名称创建了另一个配置文件：

1. 调用 IAM `GetInstanceProfile` 操作，获取 `InstanceProfileId`。
2. 调用 Amazon EC2 `DescribeInstances` 操作，获取实例的 `IamInstanceProfileId`。
3. 验证 IAM 操作的 `InstanceProfileId` 与 Amazon EC2 操作的 `IamInstanceProfileId` 是否匹配。

如果两个 ID 不同，那么附加至实例的实例配置文件将失去效力。您必须将有效的实例配置文件附加到该实例。

iam/info 文档指示成功，但指示 **"Message":"Instance Profile does not contain a role..."**。

这意味着已经通过 IAM `RemoveRoleFromInstanceProfile` 操作从实例配置文件删除了该角色。您可以使用 IAM `AddRoleToInstanceProfile` 操作为实例配置文件附加一个角色。您的应用程序需要等到下一次更新才能获得适用于该角色的凭证。

要强制进行更改，您必须取消关联实例配置文件，然后再次关联实例配置文件；您也可以停止并重新启动实例。

iam/security-credentials/[role-name] 文档指示 **"Code":"AssumeRoleUnauthorizedAccess"**

Amazon EC2 没有担任角色的许可。担任该角色的权限受制于该角色附属的信任策略，如下例所示。请使用 IAM `UpdateAssumeRolePolicy` API 更新信任策略。

JSON

```
{"Version":"2012-10-17",       "Statement": [{"Effect": "Allow","Principal":
 {"Service": ["ec2.amazonaws.com"]},"Action": ["sts:AssumeRole"]}]}
```

您的应用程序需要等到下一次自动更新才能获得适用于该角色的凭证。

要强制进行更改，您必须取消关联实例配置文件，然后再次关联实例配置文件；您也可以停止并重新启动实例。

排查 IAM 和 Amazon S3 问题

使用此处的信息可帮助您对使用 Amazon S3 和 IAM 过程中可能遇到的问题进行排查和修复。

如何授权匿名访问 Amazon S3 存储桶？

若您在 `principal` 元素中使用指定通配符的 (*) 的 Amazon S3 存储桶策略，则意味着任何人都可以访问该存储桶。通过匿名访问，任何人（包括没有 Amazon Web Services 账户 的用户）都能够访问该存储桶。有关策略示例，请参阅 Amazon Simple Storage Service 用户指南中的 Amazon S3 存储桶策略示例。

我以 Amazon Web Services 账户 根用户身份登录。为什么我无法访问自己账户下的 Amazon S3 存储桶？

在某些情况下，您可能拥有能够完全访问 IAM 和 Amazon S3 的 IAM 用户。如果 IAM 用户将存储桶策略指定给 Amazon S3 存储桶，而且未指定 根用户作为主体，则根用户访问该存储桶会被拒绝。不过，作为根用户，您仍然可以访问该存储桶。为此，请修改存储桶策略以允许根用户从 Amazon S3 控制台或 Amazon CLI 中进行访问。使用以下主体，同时将 *123456789012* 替换为 Amazon Web Services 账户 的 ID。

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

排查 SAML 与 IAM 联合身份验证的问题

使用此处的信息可帮助您诊断和修复在将 SAML 2.0 与 Amazon Identity and Access Management 联合使用时可能遇到的问题。

主题

- 错误：您的请求包含无效的 SAML 响应。要注销，请单击此处。
- 错误：AuthnResponse 中需要 RoleSessionName（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）

- 错误：未授权执行 sts:AssumeRoleWithSAML（服务：AWSSecurityTokenService；状态代码：403；错误代码：AccessDenied）
- 错误：AuthnResponse 中的 RoleSessionName 必须匹配 [a-zA-Z_0-9+=,.@-]{2,64}（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）
- 错误：源身份必须匹配 [a-zA-Z_0-9+=,.@-]{2,64} 且不能以 "aws:" 开始（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）
- 错误：响应签名无效（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）
- 错误：私钥无效。
- 错误：无法删除私有密钥。
- 错误：无法删除私有密钥，因为密钥 ID 与私有密钥不匹配。
- 错误：无法担任角色：指定提供商中没有发布者（服务：AWSOpenIdDiscoveryService；状态代码：400；错误代码：AuthSamlInvalidSamlResponseException）
- 错误：无法解析元数据。
- 错误：无法更新身份提供商。没有为元数据或加密断言定义任何更新。
- 错误：无法将断言加密模式设置为“必需”，因为未提供私有密钥。
- 错误：无法在同一请求中添加和删除私钥。仅为两个参数中的一个参数设置值。
- 错误：指定的提供商不存在。
- 错误：请求的 DurationSeconds 超过为该角色设置的 MaxSessionDuration。
- 错误：已达到私有密钥限制 2。
- 错误：响应不包含所需受众。

错误：您的请求包含无效的 SAML 响应。要注销，请单击此处。

当来自身份提供商的 SAML 响应没有包含将 `Name` 设置为 `https://aws.amazon.com/SAML/Attributes/Role` 的属性时，会出现此错误。属性必须包含一个或多个 `AttributeValue` 元素，每个元素中包含以逗号分隔的一对字符串：

- 用户可以映射到的角色的 ARN
- SAML 提供商的 ARN

当身份提供者发送的 SAML 属性值包含前导或尾随空格，或者 SAML 属性值中包含其他无效字符时，也会发生该错误。有关 SAML 属性的预期值的更多信息，请参阅为身份验证响应配置 SAML 断言。

有关更多信息，请参阅 为身份验证响应配置 SAML 断言。。要在浏览器中查看 SAML 响应，请按照在您的浏览器中查看 SAML 响应中列出的步骤操作。

错误：AuthnResponse 中需要 RoleSessionName（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）

当来自身份提供商的 SAML 响应没有包含将 Name 设置为 `https://aws.amazon.com/SAML/Attributes/RoleSessionName` 的属性时，会出现此错误。属性值是用户的标识符，通常是用户 ID 或电子邮件地址。

有关更多信息，请参阅 为身份验证响应配置 SAML 断言。。要在浏览器中查看 SAML 响应，请按照在您的浏览器中查看 SAML 响应中列出的步骤操作。

错误：未授权执行 sts:AssumeRoleWithSAML（服务：AWSSecurityTokenService；状态代码：403；错误代码：AccessDenied）

当在 SAML 响应中指定的 IAM 角色有拼写错误或者不存在时，会出现此错误。确保使用角色的确切名称，因为角色名称区分大小写。在 SAML 服务提供商配置中更正角色的名称。

只有当角色信任策略包含 `sts:AssumeRoleWithSAML` 操作时，才允许您访问。如果您的 SAML 断言配置为使用 `PrincipalTag` 属性，则信任策略还必须包含 `sts:TagSession` 操作。有关会话标签的更多信息，请参阅 在 Amazon STS 中传递会话标签。

如果角色信任策略中没有 `sts:SetSourceIdentity` 权限，则可能发生此错误。如果您的 SAML 断言配置为使用 `SourceIdentity` 属性，则信任策略还必须包含 `sts:SetSourceIdentity` 操作。更多有关源身份的信息，请参阅 监控和控制使用所担任角色执行的操作。

如果联合主体没有担任该角色的权限，则会出现此错误。该角色必须具有将 IAM SAML 身份提供商的 ARN 指定为 `Principal` 的信任策略。角色还包含控制哪些用户可以担任该角色的条件。确保您的用户满足条件要求。

如果 SAML 响应中没有包含 `Subject` 的 `NameID`，也会出现此错误。

有关更多信息，请参阅使 SAML 2.0 联合主体能够访问 Amazon Web Services 管理控制台和为身份验证响应配置 SAML 断言。。要在浏览器中查看 SAML 响应，请按照在您的浏览器中查看 SAML 响应中列出的步骤操作。

错误：AuthnResponse 中的 RoleSessionName 必须匹配 [a-zA-Z_0-9+=,.@-]{2,64}（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）

如果 `RoleSessionName` 属性值太长或者包含无效字符，则会出现此错误。最大有效长度为 64 个字符。

有关更多信息，请参阅 为身份验证响应配置 SAML 断言。。要在浏览器中查看 SAML 响应，请按照在您的浏览器中查看 SAML 响应中列出的步骤操作。

错误：源身份必须匹配 [a-zA-Z_0-9+=,.@-]{2,64} 且不能以 **"aws:"** 开始（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）

如果 `sourceIdentity` 属性值太长或者包含无效字符，则会出现此错误。最大有效长度为 64 个字符。更多有关源身份的信息，请参阅 监控和控制使用所担任角色执行的操作。

有关创建 SAML 断言的更多信息，请参阅 为身份验证响应配置 SAML 断言。。要在浏览器中查看 SAML 响应，请按照在您的浏览器中查看 SAML 响应中列出的步骤操作。

错误：响应签名无效（服务：AWSSecurityTokenService；状态代码：400；错误代码：InvalidIdentityToken）

在身份提供商的联合身份元数据与 IAM 身份提供商的元数据不匹配时，会出现此错误。例如，身份服务提供商的元数据文件可能已更改，以便更新过期的证书。从身份服务提供商下载更新后的 SAML 元数据文件。然后在 Amazon 身份提供商实体中更新它，该实体在 IAM 中使用 `aws iam update-saml-provider` 跨平台 CLI 命令或 `Update-IAMSAMLProvider` PowerShell cmdlet 定义。

错误：私钥无效。

如果您的私钥文件格式不正确，可能会发生此错误。此错误可能会提供有关私钥无效原因的其他详细信息：

- 密钥已加密。
- 无法识别密钥格式。私钥文件必须是 .pem 文件。

在 Amazon Web Services 管理控制台 中 在 IAM 中创建 SAML 身份提供者 时，必须从身份提供商处下载私有密钥以提供给 IAM 来启用加密。私有密钥必须是使用 AES-GCM 或 AES-CBC 加密算法解密 SAML 断言的 .pem 文件。

错误：无法删除私有密钥。

当 SAML 加密设置为“必需”时，可能会发生此错误，并且您的请求会删除 IAM SAML 提供商的唯一私有解密密钥。有关轮换私有密钥的更多信息，请参阅 管理 SAML 加密密钥。

错误：无法删除私有密钥，因为密钥 ID 与私有密钥不匹配。

如果私有密钥的 `keyId` 值与身份提供商的私有密钥文件的任一密钥 ID 不匹配，则可能发生此错误。

当您使用 update-saml-provider 或 UpdateSAMLProvider API 操作删除 SAML 加密私有密钥时，`RemovePrivateKey` 中的值必须是附加到您的身份提供商的私有密钥文件的有效密钥 ID。

错误：无法担任角色：指定提供商中没有发布者（服务：AWSOpenIdDiscoveryService；状态代码：400；错误代码：AuthSamlInvalidSamlResponseException）

如果 SAML 响应中的颁发者与联合元数据文件中声明的颁发者不匹配，则可能会出现该错误。在 IAM 中创建身份提供商时，元数据文件将上传到 Amazon 中。

错误：无法解析元数据。

如果您的元数据文件没有正确格式化，则可能发生此错误。

在 Amazon Web Services 管理控制台 中创建或管理 SAML 身份提供商时，您必须从您的身份提供商处检索 SAML 元数据文档。

此元数据文件包括颁发者名称、过期信息以及可用来验证从 IdP 处收到的 SAML 身份验证响应 (断言) 的密钥。元数据文件必须采用不含字节顺序标记 (BOM) 的 UTF-8 格式编码。要删除 BOM，您可以使用 Notepad++ 等文本编辑工具以 UTF-8 格式对文件进行编码。

作为 SAML 元数据文档的一部分，X.509 证书必须使用长度至少为 1024 位的密钥。此外，X.509 证书也不能有任何重复的扩展名。您可以使用扩展程序，但扩展程序只能在证书中显示一次。如果 X.509 证书不符合任一条件，则 IdP 将创建失败，并返回“Unable to parse metadata”这一错误消息。

如 SAML V2.0 元数据互操作性配置文件 1.0 版所定义，IAM 既不会评估 SAML 元数据文档的 X.509 证书，也不会在该证书过期时采取任何行动。如果您担心 X.509 证书过期，建议您监控证书到期日期，并根据贵组织的治理和安全策略来轮换证书。

错误：无法更新身份提供商。没有为元数据或加密断言定义任何更新。

如果使用 `update-saml-provider` CLI 或 `UpdateSAMLProvider` API 操作，但未在请求参数中提供更新值，则可能会发生此错误。有关更新 IAM SAML 提供商的更多信息，请参阅 在 IAM 中创建 SAML 身份提供者。

错误：无法将断言加密模式设置为“必需”，因为未提供私有密钥。

如果之前未上传私有解密密钥，并且尝试将 SAML 加密设置为“必需”，但请求中未包含私有密钥，则可能会发生此错误。

在使用 `create-saml-provider` CLI、`CreateSAMLProvider` API、`update-saml-provider` CLI 或 `UpdateSAMLProvider` API 操作要求加密的 SAML 断言时，请确保为 IAM SAML 提供商定义了私有密钥。

错误：无法在同一请求中添加和删除私钥。仅为两个参数中的一个参数设置值。

如果添加和删除私钥值都包含在同一请求中，则可能会发生此错误。

当您使用 update-saml-provider 或 UpdateSAMLProvider API 操作来轮换 SAML 加密私有密钥文件时，您只能在请求中添加或删除私有密钥。如果在删除私有密钥的同时添加私有密钥，则操作会失败。有关轮换私有密钥的更多信息，请参阅 管理 SAML 加密密钥。

错误：指定的提供商不存在。

如果 SAML 断言中的提供商名称与 IAM 中的提供商名称不匹配，则可能会出现该错误。有关查看提供商名称的更多信息，请参阅在 IAM 中创建 SAML 身份提供者。

错误：请求的 DurationSeconds 超过为该角色设置的 MaxSessionDuration。

如果从 Amazon CLI 或 API 中担任角色，则可能会出现该错误。

在使用 assume-role-with-saml CLI 或 AssumeRoleWithSAML API 操作担任角色时，您可以为 `DurationSeconds` 参数指定一个值。您可以指定 900 秒 (15 分钟) 到角色的最大会话持续时间设置之间的值。如果指定的值高于该设置，操作将失败。例如，如果您指定的会话持续时间为 12 小时，但管理员设置的最大会话持续时间为 6 小时，您的操作将失败。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。

错误：已达到私有密钥限制 2。

如果您试图将私有密钥添加到身份提供商，则可能会发生此错误。

每个身份提供商最多可以保存两个私有密钥。当您使用 update-saml-provider 或 UpdateSAMLProvider API 操作来添加第三个私有密钥时，操作会失败。

删除已过期的私有密钥，然后再添加新的私有密钥。有关轮换私有密钥的更多信息，请参阅 管理 SAML 加密密钥。

错误：响应不包含所需受众。

如果受众 URL 与 SAML 配置中的身份提供者不匹配，则可能会出现此错误。确保您的身份提供者（IdP）依赖方标识符与 SAML 配置中提供的受众 URL（实体 ID）完全匹配。

IAM 如何与其他 Amazon 服务协同工作

虽然 IAM 是您用来管理 IAM 资源的主要 Amazon 服务，但所有其他 Amazon 服务都可以与 IAM 配合使用来控制对您账户中资源的访问权限。

- Amazon CloudFormation

  Amazon CloudFormation 与 IAM 集成，允许您将 IAM 资源作为 Amazon CloudFormation 模板的一部分进行定义和管理。您可以使用 Amazon CloudFormation 为您预置的其他 Amazon 资源指定必要的 IAM 权限。Amazon CloudFormation 还支持使用 IAM 角色来管理预置和管理 Amazon 基础设施所需的凭证，其偏差检测功能可帮助您维护 IAM 配置的完整性。

- Amazon CloudShell

  当您访问 Amazon CloudShell 时，您的身份验证和授权将通过 IAM 进行处理。Amazon CloudShell 在分配给您的用户或账户的 IAM 角色上下文中运行。启动 Amazon CloudShell 时，它会根据分配给您的 IAM 角色自动生成临时安全凭证。

- Amazon SDK

  Amazon SDK 与 IAM 配合使用，处理身份验证和授权流程、管理 Amazon 凭证并遵守 IAM 中定义的权限和策略，以确保您的应用程序只能访问其获得授权使用的资源。这些 SDK 提供了获取和使用临时安全凭证以及验证应用程序操作所需权限的机制。

有关与 IAM 配合使用的 Amazon 服务列表以及这些服务支持的 IAM 功能，请参阅 使用 IAM 的 Amazon 服务。

使用 Amazon CloudFormation 创建 IAM 资源

Amazon Identity and Access Management 与 Amazon CloudFormation 集成，后者是一项服务，可帮助您对 Amazon 资源进行建模和设置，这样您只需花较少的时间来创建和管理资源与基础设施。您可以创建一个模板来描述所需的所有 Amazon 资源 [例如访问密钥、组、组策略、实例配置文件、托管式策略、OIDC 提供者、内联策略、角色、角色策略、SAML 提供者、服务器证书、服务相关角色、用户（以及向组添加用户）、用户策略和虚拟 MFA 设备等]，然后 Amazon CloudFormation 将为您预置和配置这些资源。

如果使用 Amazon CloudFormation，则可以重复使用您的模板来重复设置您的 IAM 资源并确保一致性。描述您的资源一次，然后在多个 Amazon Web Services 账户和区域中反复预调配相同的资源。

IAM 和 Amazon CloudFormation 模板

要预置和配置 IAM 和相关服务的资源，您必须了解 Amazon CloudFormation 模板。模板是 JSON 或 YAML 格式的文本文件。这些模板可描述您要在 Amazon CloudFormation 堆栈中预调配的资源。如果您不熟悉 JSON 或 YAML，可以在 Amazon CloudFormation Designer 的帮助下开始使用 Amazon CloudFormation 模板。有关更多信息，请参阅《Amazon CloudFormation 用户指南》中的什么是 Amazon CloudFormation Designer？。

IAM 支持在 Amazon CloudFormation 中创建访问密钥、组、组策略、实例配置文件、托管式策略、OIDC 提供者、内联策略、角色、角色策略、SAML 提供者、服务器证书、服务相关角色、用户（以及向组添加用户）、用户策略和虚拟 MFA 设备等。有关更多信息，包括 IAM 资源的 JSON 和 YAML 模板示例，请参阅《Amazon CloudFormation 用户指南》中的 Amazon Identity and Access Management resource type reference。

您也可以通过创建模板来创建相关资源，例如角色和托管式策略。

了解有关 Amazon CloudFormation 的更多信息

要了解有关 Amazon CloudFormation 的更多信息，请参阅以下资源：

- Amazon CloudFormation
- 《Amazon CloudFormation 用户指南》
- Amazon CloudFormation API 参考
- 《Amazon CloudFormation 命令行界面用户指南》

将 Amazon CloudShell 与 Amazon Identity and Access Management 结合使用

Amazon CloudShell 是一个已经事先完成身份验证的浏览器式 Shell，您可以直接从Amazon Web Services 管理控制台启动它。您可以使用自己惯用的 Shell（Bash、PowerShell 或 Z shell），针对 Amazon 服务（包括 Amazon Identity and Access Management）运行 Amazon CLI 命令。您无需下载或安装命令行工具，即可完成此操作。

您可以 从 Amazon Web Services 管理控制台 启动 Amazon CloudShell，用于登录控制台的 Amazon 凭证将在新的 Shell 会话中自动可用。通过这种对 Amazon CloudShell 用户进行预身份验证，使您可在使用 Amazon CLI 版本 2（在 Shell 的计算环境中预装）与 IAM 等 Amazon 服务进行交互时跳过凭证配置步骤。

获取 Amazon CloudShell 的 IAM 权限

使用 Amazon Identity and Access Management 提供的访问管理资源，管理员可以向 IAM 用户授予权限，使其能够访问 Amazon CloudShell 并使用环境的功能。

管理员要向用户授予访问权限，最快捷的方法是通过 Amazon 托管式策略。Amazon 托管式策略是由 Amazon 创建和管理的独立策略。可以将以下适用于 CloudShell 的 Amazon 托管式策略附加到 IAM 身份：

- `AWSCloudShellFullAccess`：授予使用 Amazon CloudShell 的权限，并具有对所有功能的完全访问权限。

如果要限制 IAM 用户可以使用 Amazon CloudShell 执行的操作范围，则可以 `AWSCloudShellFullAccess` 托管式策略为模板创建使用的定义策略。要详细了解如何限制用户可在 CloudShell 中使用的操作，请参阅《Amazon CloudShell 用户指南》中的 Managing Amazon CloudShell access and usage with IAM policies。

与 IAM 交互

从 Amazon Web Services 管理控制台 启动 Amazon CloudShell 后，您可以立即开始使用命令行界面与 IAM 进行交互。

> **Note**
>
> 在 Amazon CloudShell 中使用 Amazon CLI 时，无需下载或安装任何其他资源。此外，由于已经在 Shell 中进行了身份验证，因此在进行调用之前无需配置凭证。

使用 Amazon CloudShell 创建 IAM 组并将 IAM 用户添加到该组

以下示例使用 CloudShell 创建了一个 IAM 组，向该组添加了一个 IAM 用户，然后验证命令是否成功。

1. 在 Amazon Web Services 管理控制台 中，您可以选择导航栏中提供的下列可用选项来启动 CloudShell：
   - 选择 CloudShell 图标。
   - 首先在搜索框中键入“cloudshell”，然后选择 CloudShell 选项。

2. 要创建 IAM 组，请在 CloudShell 命令行中输入以下命令。在此例中，我们将该组命名为 east_coast：

```
aws iam create-group --group-name east_coast
```

如果调用成功，命令行将显示来自服务的响应，输出与以下类似：

```
{
   "Group": {
       "Path": "/",
       "GroupName": "east_coast",
       "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
       "Arn": "arn:aws:iam::111122223333:group/east_coast",
       "CreateDate": "2023-09-11T21:02:21+00:00"
    }
}
```

3. 要将用户添加到您创建的组，请使用以下命令，并指定组名和用户名。在此例中，我们将该组命名为 east_coast，将用户命名为 john：

```
aws iam add-user-to-group --group-name east_coast --user-name john
```

4. 要验证用户是否在该组中，请使用以下命令并指定组名。在此例中，我们继续使用组 east_coast：

```
aws iam get-group --group-name east_coast
```

如果调用成功，命令行将显示来自服务的响应，输出与以下类似：

```
{
  "Users": [
    {
        "Path": "/",
        "UserName": "john",
        "UserId": "AIDAYBDBW4JBXGEXAMPLE",
        "Arn": "arn:aws:iam::552108220995:user/john",
        "CreateDate": "2023-09-11T20:43:14+00:00",
        "PasswordLastUsed": "2023-09-11T20:59:14+00:00"
```

```
        }
      ],
      "Group": {
            "Path": "/",
            "GroupName": "east_coast",
            "GroupId": "AGPAYBDBW4JBY3EXAMPLE",
            "Arn": "arn:aws:iam::111122223333:group/east_coast",
            "CreateDate": "2023-09-11T21:02:21+00:00"
        }
    }
```

将此服务与 Amazon SDK 结合使用

Amazon 软件开发工具包（SDK）适用于许多常用编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档
Amazon CLI
适用于 Java 的 Amazon SDK
适用于 JavaScript 的 Amazon SDK
适用于 .NET 的 Amazon SDK
适用于 PHP 的 Amazon SDK
Amazon Tools for PowerShell
适用于 Python (Boto3) 的 Amazon SDK
适用于 Ruby 的 Amazon SDK
适用于 SAP ABAP 的 Amazon SDK

有关特定于此服务的示例，请参阅使用 Amazon 开发工具包的 IAM 代码示例。

Amazon Identity and Access Management 的参考信息

使用本节中的主题查找 IAM 和 Amazon STS 各方面的详细参考资料。

主题

- 使用 Amazon 资源名称（ARN）标识 Amazon 资源
- IAM 标识符
- IAM 和 Amazon STS 配额
- 双堆栈端点支持
- 接口 VPC 端点
- 使用 IAM 的 Amazon 服务
- 适用于 API 请求的 Amazon 签名版本 4
- IAM JSON 策略参考

- 用于编程访问的简化 Amazon Web Services 服务信息

使用 Amazon 资源名称（ARN）标识 Amazon 资源

Amazon Resource Name（ARN）唯一标识 Amazon 资源。当您需要在 Amazon 全局环境中（比如 IAM policy、Amazon Relational Database Service (Amazon RDS) 标签和 API 调用中）明确指定一项资源时，我们要求使用 ARN。虽然应像任何识别信息一样谨慎使用和共享 ARN，但不应将其视为机密、敏感或保密信息。

ARN 格式

以下是 ARN 的一般格式。特定格式取决于资源。要使用 ARN，请将*##* 文本替换为特定于资源的信息。请注意，某些资源的 ARN 忽略了区域、账户 ID 或同时忽略了这两者。

```
arn:partition:service:region:account-id:resource-id
arn:partition:service:region:account-id:resource-type/resource-id
arn:partition:service:region:account-id:resource-type:resource-id
```

partition

资源所在的分区。分区 是一组 Amazon 区域。每个 Amazon 账户的作用域为一个分区。

以下是支持的分区：

- aws - Amazon 区域
- aws-cn – 中国区域
- aws-us-gov - Amazon GovCloud (US) 区域

service

标识 Amazon 产品的服务命名空间。

region

区域代码。

account-id

拥有资源的 Amazon 账户的 ID（不含连字符）。例如 123456789012。

resource-type

资源类型。例如，虚拟私有云（VPC）的 vpc。

resource-id

资源标识符。这是资源的名称、资源的 ID 或资源路径。某些资源标识符包括父资源 (sub-resource-type/parent-resource/sub-resource) 或限定符（例如版本）(resource-type:resource-name:qualifier)。

示例

IAM 用户

arn:aws:iam::*123456789012*:user/*john*

SNS 主题

arn:aws:sns:*us-east-1*:*123456789012*:*example-sns-topic-name*

VPC

arn:aws:ec2:*us-east-1*:*123456789012*:vpc/*vpc-0e9801d129EXAMPLE*

查找资源的 ARN 格式

ARN 的具体格式取决于服务和资源类型。某些资源 ARN 可以包含路径、变量或通配符。如需查找特定 Amazon 资源的 ARN 格式，请打开服务授权参考，然后打开该服务的页面，并导航至资源类型表。

ARN 中的路径

资源 ARN 可以包含路径。例如，在 Amazon S3 中，资源标识符是一个对象名称，它可以包含正斜杠（/）来形成路径。同样，IAM 用户名称和组名也可以包含路径。IAM 路径中只允许使用字母数字字符和以下字符：正斜杠（/）、加号（+）、等号（=）、英文逗号（,）、英文句号（.）、at 符（@）、下划线（_）和连字符（-）。

在路径中使用通配符

路径可以包含一个通配符，即星号 (*)。某些策略元素允许使用通配符，而其他元素则不允许。您可以对 Resource 或 NotResource 元素使用通配符，但不能对 Principal 或 NotPrincipal 元素使用通配符。有关更多信息，请参阅 IAM JSON 策略参考。

您可以指定 `role/*` 来表示账户 123456789012 中的所有角色，如下例所示：

```
arn:aws:iam::123456789012:role/*
```

您也可以使用通配符结束资源名称。例如，您可以指定 **`service-*`** 来表示所有以 `service` 开头并以不同字符结尾的角色，如 `service-role1` 或 `service-test`：

```
arn:aws:iam::123456789012:role/service-*
```

以下示例显示了 Amazon S3 存储桶中的对象的 ARN，其中的资源名称包含一个路径。ARN `arn:aws:s3:::amzn-s3-demo-bucket/*` 适用于该存储桶内的所有对象，无论前缀是什么。ARN `arn:aws:s3:::amzn-s3-demo-bucket/Development/*` 适用于在 **`/Development/`** 前缀内创建的所有对象。

您还可以使用 ? 通配符来指定 ARN 中的一个字符。例如，您可以将以下 ARN 用于名为 amzn-s3-demo-bucket 的 S3 存储桶中所有以四个字符开头并以 **`-test`** 结尾的文件夹。与此匹配的一些文件夹包括 `1234-test`、`2024-test` 或 `a100-test`。

```
arn:aws:s3:::amzn-s3-demo-bucket/????-test
```

您还可以在 ARN 的不同部分中使用通配符，以冒号“:”分隔。在以下示例中，两个通配符用于匹配账户 123456789012 的所有区域中的所有 Amazon Q 应用程序及应用程序中的资源：

```
arn:aws:qbusiness:*:123456789012:*
```

类似地，以下示例匹配账户 123456789012 的所有区域中的所有 Amazon VPC：

```
arn:aws:ec2:*:123456789012:vpc/*
```

以下示例匹配账户 123456789012 的所有区域中的所有 Amazon EBS 卷：

```
arn:aws:ec2:*:123456789012:volume/*
```

ARN 中通配符使用的限制

您不能在指定资源类型的 ARN 部分中使用通配符。以下资源类型中带有通配符的示例 ARN 无效：

```
arn:aws:lambda:us-east-2:123456789012:functi*:my-function <== not allowed
```

> **Note**
>
> 当您在基于身份的策略中指定不完整的 ARN（少于标准的六个字段）时，Amazon 会通过在所有缺失字段中添加通配符 (*) 来自动完成 ARN。例如，指定 `arn:aws:sqs` 相当于 `arn:aws:sqs:*:*:*`，它让您可以访问所有区域和账户中的所有 Amazon SQS 资源。但是，传递给 Amazon STS AssumeRole、AssumeRoleWithWebIdentity 和 AssumeRoleWithSAML 请求的会话策略不支持不完整的 ARN。在会话策略中使用不完整的 ARN 会导致 `MalformedPolicyDocumentException` 错误。

您也不能在前缀 ARN 中使用通配符，也不能在 ARN 的分区部分中使用通配符。

```
arn:aws:redshift:us-east-1:123456789012:? <== not allowed
```

IAM 标识符

IAM 针对用户、IAM 组、角色、策略及服务器证书使用多个不同的标识符。本部分主要介绍标识符及您何时使用每个标识符。

主题

- 易记名称和路径
- IAM ARN
- 唯一标识符
- 了解唯一 ID 前缀
- 获取唯一标识符

易记名称和路径

在您创建用户、角色、用户组或策略时，或在您上传服务器证书时，您可为其取一个易于识别的名称。例如 Bob、TestApp1、Developers、ManageCredentialsPermissions 或 ProdServerCert。

如果您使用 IAM API 或 Amazon Command Line Interface（Amazon CLI）创建 IAM 资源，则可以添加可选路径。您可以使用单个路径，或嵌套多个路径，就像它们是文件夹结构一样。例如，可使用嵌套路径 `/division_abc/subdivision_xyz/product_1234/engineering/` 以匹配贵公司的企业结构。您可以随后创建策略以允许该路径的所有用户访问策略模拟器 API。要查看该策略，请参阅 IAM：基于用户路径访问策略模拟器 API。有关如何指定友好名称的信息，请参阅用户 API 文档。有关如何使用路径的其他示例，请参阅 IAM ARN。

使用 Amazon CloudFormation 创建资源时，您可以为用户、IAM 组和角色以及客户管理型策略指定路径。

如果您的用户和用户组位于同一路径中，IAM 不会自动将该用户置于该用户组中。例如，您可创建一个 Developers 组，并指定其路径为 `/division_abc/subdivision_xyz/product_1234/engineering/`。如果您创建了一个名为 Bob 的用户并向其添加了相同的路径，则这不会自动将 Bob 归入开发人员用户组。IAM 不会基于路径而强行在用户或 IAM 组之间划分边界。具有不同路径的用户可以使用相同的资源（假设他们已获权使用这些资源）。Amazon 账户中 IAM 资源的数量和大小是有限的。有关更多信息，请参阅 IAM 和 Amazon STS 配额。

IAM ARN

大多数资源都有易记名称（例如，名为 Bob 的用户或名为 Developers 的用户组）。不过，权限策略语言要求您使用以下 Amazon Resource Name (ARN) 格式指定资源。

```
arn:partition:service:region:account:resource
```

其中：

- `partition` 用于标识资源所在的分区。对于标准 Amazon 区域，分区是 `aws`。如果资源位于其他分区，则分区是 `aws-`*`partitionname`*。例如，中国（北京）区域中的资源的分区为 `aws-cn`。您不能在不同分区的账户之间委派访问权限。
- `service` 标识 Amazon 产品。IAM 资源始终使用 `iam`。
- `region` 标识资源的区域。对于 IAM 资源，它始终保持空白。
- `account` 指定没有连字符的 Amazon Web Services 账户 ID。

- `resource` 按名称标识特定资源。

您可以使用以下语法指定 IAM 和 Amazon STS ARN。由于 IAM 资源是全球资源，因此，ARN 的区域部分是空的。

语法：

```
arn:aws:iam::account:root
arn:aws:iam::account:user/user-name-with-path
arn:aws:iam::account:group/group-name-with-path
arn:aws:iam::account:role/role-name-with-path
arn:aws:iam::account:policy/policy-name-with-path
arn:aws:iam::account:instance-profile/instance-profile-name-with-path
arn:aws:sts::account:federated-user/user-name
arn:aws:sts::account:assumed-role/role-name/role-session-name
arn:aws:sts::account:self
arn:aws:iam::account:mfa/virtual-device-name-with-path
arn:aws:iam::account:u2f/u2f-token-id
arn:aws:iam::account:server-certificate/certificate-name-with-path
arn:aws:iam::account:saml-provider/provider-name
arn:aws:iam::account:oidc-provider/provider-name
arn:aws:iam::aws:contextProvider/context-provider-name
```

以下示例中有很多在 ARN 的资源部分中包含路径。不能在 Amazon Web Services 管理控制台中创建或操作路径。要使用路径，必须通过使用 Amazon API、Amazon CLI 或 Tools for Windows PowerShell 与资源相结合。

示例：

```
arn:aws:iam::123456789012:root
arn:aws:iam::123456789012:user/John
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Jane
arn:aws:iam::123456789012:group/Developers
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
arn:aws:iam::123456789012:role/S3Access
arn:aws:iam::123456789012:role/application_abc/component_xyz/RDSAccess
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
arn:aws:iam::123456789012:role/service-role/QuickSightAction
arn:aws:iam::123456789012:policy/UsersManageOwnCredentials
arn:aws:iam::123456789012:policy/division_abc/subdivision_xyz/UsersManageOwnCredentials
```

```
arn:aws:iam::123456789012:instance-profile/Webserver
arn:aws:sts::123456789012:federated-user/John
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Jane
arn:aws:sts::123456789012:self
arn:aws:iam::123456789012:mfa/JaneMFA
arn:aws:iam::123456789012:u2f/user/John/default (U2F security key)
arn:aws:iam::123456789012:server-certificate/ProdServerCert
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
arn:aws:iam::123456789012:oidc-provider/server.example.org
arn:aws:iam::aws:contextProvider/IdentityCenter
```

以下示例提供了更详细的信息，帮助您了解不同类型 IAM 和 Amazon STS 资源的 ARN 格式。

- 账户中的 IAM 用户：

> **Note**
>
> 每个 IAM 用户名都是唯一的。用户的用户名不区分大小写，例如在登录过程中，但在策略或 ARN 中使用时区分大小写。

```
arn:aws:iam::123456789012:user/John
```

- 具有反映组织结构图的路径的其他用户：

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/Jane
```

- 对于 IAM 用户组：

```
arn:aws:iam::123456789012:group/Developers
```

- 具有路径的 IAM 用户组：

```
arn:aws:iam::123456789012:group/division_abc/subdivision_xyz/product_A/Developers
```

- IAM 角色：

```
arn:aws:iam::123456789012:role/S3Access
```

- 服务相关角色：

```
arn:aws:iam::123456789012:role/aws-service-role/access-analyzer.amazonaws.com/
AWSServiceRoleForAccessAnalyzer
```

- 服务角色：

```
arn:aws:iam::123456789012:role/service-role/QuickSightAction
```

- 托管策略：

```
arn:aws:iam::123456789012:policy/ManageCredentialsPermissions
```

- 可与 Amazon EC2 实例关联的实例配置文件：

```
arn:aws:iam::123456789012:instance-profile/Webserver
```

- 在 IAM 中识别为“Paulo”的 Amazon STS 联合用户主体：

```
arn:aws:sts::123456789012:federated-user/Paulo
```

- 担任“Accounting-Role”且角色会话名称为“Mary”的人的活动会话：

```
arn:aws:sts::123456789012:assumed-role/Accounting-Role/Mary
```

- 表示在调用会话上运行的 API 调用（例如 Amazon STS SetContext API）中用作资源时调用方自己的会话：

```
arn:aws:sts::123456789012:self
```

- 分配给名为 Jorge 的用户的多重身份验证设备：

```
arn:aws:iam::123456789012:mfa/Jorge
```

- 服务器证书：

```
arn:aws:iam::123456789012:server-certificate/ProdServerCert
```

- 具有反映组织结构图的路径的服务器证书：

```
arn:aws:iam::123456789012:server-certificate/division_abc/subdivision_xyz/
ProdServerCert
```

- 身份提供程序 (SAML 和 OIDC)：

```
arn:aws:iam::123456789012:saml-provider/ADFSProvider
arn:aws:iam::123456789012:oidc-provider/GoogleProvider
arn:aws:iam::123456789012:oidc-provider/server.example.org
```

- OIDC 身份提供者，其路径反映了某个 Amazon EKS OIDC 身份提供者的 URL：

```
arn:aws:iam::123456789012:oidc-provider/oidc.eks.us-west-2.amazonaws.com/id/
a1b2c3d4567890abcdefEXAMPLE11111
```

- 用于 Amazon IAM Identity Center 可信身份传播的上下文提供程序 ARN，可信上下文断言就是从该 ARN 生成的：

```
arn:aws:iam::aws:contextProvider/IdentityCenter
```

另一个重要的 ARN 是根用户 ARN。虽然这不是一个 IAM 资源，但您应熟悉此 ARN 的格式。它通常用于基于资源的策略的 Principal 元素。

- Amazon Web Services 账户 显示以下内容：

```
arn:aws:iam::123456789012:root
```

以下示例说明了您可向 Richard 分配的策略，该策略允许其管理访问密钥。请注意，这里的资源即是 IAM 用户 Richard。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ManageRichardAccessKeys",
            "Effect": "Allow",
            "Action": [
```

```
                "iam:*AccessKey*",
                "iam:GetUser"
            ],
            "Resource": "arn:aws:iam::*:user/division_abc/subdivision_xyz/
Richard"
        },
        {
            "Sid": "ListForConsole",
            "Effect": "Allow",
            "Action": "iam:ListUsers",
            "Resource": "*"
        }
    ]
}
```

> **Note**
>
> 使用 ARN 在 IAM policy 中标识资源时，您可以包含策略变量。策略变量可以包含表示运行时信息（如用户名称）的占位符，作为 ARN 的一部分。有关更多信息，请参阅 IAM policy 元素：变量和标签

在 ARN 中使用通配符和路径

您可以在 ARN 的 *resource* 部分中使用通配符来指定多个用户、IAM 组或策略。例如，如要指定使用 product_1234 的所有用户，您应使用：

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/*
```

如果您的用户名称以字符串 app_ 开头，可以通过以下 ARN 引用所有这些用户。

```
arn:aws:iam::123456789012:user/division_abc/subdivision_xyz/product_1234/app_*
```

要指定您 Amazon Web Services 账户 中的所有用户、IAM 组或策略，请分别在 ARN 的 user/、group/ 或 policy/ 部分之后使用通配符。

```
arn:aws:iam::123456789012:user/*
arn:aws:iam::123456789012:group/*
arn:aws:iam::123456789012:policy/*
```

如果为用户 arn:aws:iam::111122223333:user/* 指定以下 ARN，则它与以下两个示例相匹配。

```
arn:aws:iam::111122223333:user/John
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/JaneDoe
```

但是，如果您为用户 arn:aws:iam::111122223333:user/division_abc* 指定以下 ARN，则它匹配第二个示例，但不匹配第一个示例。

```
arn:aws:iam::111122223333:user/John
arn:aws:iam::111122223333:user/division_abc/subdivision_xyz/Jane
```

请勿在 ARN 的 user/、group/ 或 policy/ 部分中使用通配符。例如，IAM 不允许执行以下操作：

```
arn:aws:iam::123456789012:u*
```

> **Note**
>
> 当您在基于身份的策略中指定不完整的 ARN（少于标准的六个字段）时，Amazon 会通过在所有缺失字段中添加通配符 (*) 来自动完成 ARN。例如，指定 arn:aws:sqs 相当于 arn:aws:sqs:*:*:*，它让您可以访问所有区域和账户中的所有 Amazon SQS 资源。但是，传递给 Amazon STS AssumeRole、AssumeRoleWithWebIdentity 和 AssumeRoleWithSAML 请求的会话策略不支持不完整的 ARN。在会话策略中使用不完整的 ARN 会导致 MalformedPolicyDocumentException 错误。

Example针对基于项目的用户组使用路径和 ARN 的示例

不能在 Amazon Web Services 管理控制台中创建或操作路径。要使用路径，必须通过使用 Amazon API、Amazon CLI 或 Tools for Windows PowerShell 与资源相结合。

在此示例中，Marketing_Admin 用户组中的 Jules 在 /marketing/ 路径内创建一个基于项目的用户组。Jules 并将公司内不同部门的用户分配至该用户组。此示例说明，用户的路径与用户所在的用户组并无关系。

市场部组负责一款即将上市的新产品，因此 Jules 在 /marketing/ 路径内创建了一个新的用户组，并命名为 Widget_Launch。随后，Jules 将下列策略分配至此用户组，而该策略将允许此用户组访问 example_bucket（专用于此款新产品）部分中的对象。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": "arn:aws:s3:::example_bucket/marketing/newproductlaunch/widget/
*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket*",
      "Resource": "arn:aws:s3:::example_bucket",
      "Condition": {"StringLike": {"s3:prefix": "marketing/newproductlaunch/
widget/*"}}
    }
  ]
}
```

最后，Jules 将负责此款新产品的用户分配至该用户组。其中包括 /marketing/ 路径中的 Patricia 和 Eli。另外，还包括 /sales/ 路径中的 Chris 和 Chloe，及 /legal/ 路径中的 Alice 和 Jim。

唯一标识符

当 IAM 创建用户、用户组、角色、策略、实例配置文件或服务器证书时，它会为每个资源分配一个唯一 ID。该唯一 ID 如下所示：

AIDAJQABLZS4A3QDU576Q

大多数情况下，在使用 IAM 资源时，您使用易于识别的名称和 ARN。这样，您就不需要知道特定资源的唯一 ID。不过有时，当实际情况不允许使用好友名称时，唯一 ID 可能非常有用。

有一个示例涉及到对您的 Amazon Web Services 账户 中的易记名称的重复使用。在您的账户中，用户、用户组、角色或策略的易记名称必须是唯一的。例如，您可以创建一个名为 John 的 IAM 用户。公司使用 Amazon S3 并有一个存储桶，其中包含每位员工的文件夹。IAM 用户 John 是名为 User-S3-Access 的 IAM 用户组的成员，该用户组的权限仅允许用户访问存储桶中其自己的文件夹。有关可以通过何种方式创建基于身份的策略以允许 IAM 用户使用其易记名称访问 S3 中其自己的存储桶对象的示例，请参阅 Amazon S3：允许 IAM 用户以编程方式和在控制台中访问其 S3 主目录。

假设名为 John 的员工从贵公司离职，并且您删除了名为 John 的相应 IAM 用户。但后来另一位名为 John 的员工入职贵公司，并且您创建了一个名为 John 的新 IAM 用户。您将这位名为 John 的新 IAM 用户添加到了现有 IAM 用户组 User-S3-Access 中。如果与该用户组关联的策略指定易记的 IAM 用户名 John，则该策略将允许新 John 访问前一个 John 留下的信息。

一般情况下，建议您为策略中的资源指定 ARN，而不是其唯一 ID。但是，每位 IAM 用户都具有一个唯一 ID，即使您创建的新 IAM 用户重复使用之前已删除的易记名称，也是如此。在该示例中，旧 IAM 用户 John 和新 IAM 用户 John 具有不同的唯一 ID。您可以创建基于资源的策略，这些策略按唯一 ID 而不仅仅是用户名授予访问权限。这样做可以减少您无意中向员工授予了不应具备的信息访问权限的机会。

以下示例说明了如何在基于资源策略的 Principal 元素中指定唯一 ID。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::111122223333:role/role-name",
    "AIDACKCEVSQ6C2EXAMPLE",
    "AROADBQP57FF2AEXAMPLE"
  }
```

以下示例说明了如何在策略的 Condition 元素中使用全局条件键 aws:userid 指定唯一 ID。

```
"Condition": {
    "StringLike": {
      "aws:userId": [
        "AIDACKCEVSQ6C2EXAMPLE",
        "AROADBQP57FF2AEXAMPLE:role-session-name",
        "AROA1234567890EXAMPLE:*",
        "111122223333"
      ]
    }
  }
```

用户 ID 发挥作用的另一个示例是您维护自己的 IAM 用户或角色信息数据库（或其他存储）。唯一 ID 可为您创建的每个 IAM 用户或角色提供唯一标识符。当您有 IAM 用户或角色重复使用某个名称时也是如此，如前面的示例所示。

了解唯一 ID 前缀

IAM 使用以下前缀来指示每个唯一 ID 适用于的资源的类型。前缀可能会根据创建时间的不同而有所不同。

Prefix	资源类型
ABIA	Amazon STS 服务持有者令牌
ACCA	上下文特定凭证
AGPA	用户组
AIDA	IAM 用户
AIPA	Amazon EC2 实例配置文件
AKIA	访问密钥
ANPA	托管式策略
ANVA	托管策略中的版本
APKA	公有密钥
AROA	角色
ASCA	证书
ASIA	临时 (Amazon STS) 访问密钥 ID 使用此前缀，但仅在与秘密访问密钥和会话令牌结合使用时才具有唯一性。

获取唯一标识符

无法通过 IAM 控制台获得 IAM 资源的唯一 ID。要获得唯一 ID，您可以使用下面的 Amazon CLI 命令或 IAM API 调用。

Amazon CLI:

- get-caller-identity
- get-group
- get-role
- get-user

- get-policy
- get-instance-profile
- get-server-certificate

IAM API：

- GetCallerIdentity
- GetGroup
- GetRole
- GetUser
- GetPolicy
- GetInstanceProfile
- GetServerCertificate

IAM 和 Amazon STS 配额

Amazon Identity and Access Management (IAM) 和 Amazon Security Token Service (STS) 具有限制对象大小的配额。这会影响您命名对象的方式、可以创建的对象数量以及传递对象时可以使用的字符数。

> **Note**
>
> 要获取有关 IAM 使用量和配额的账户级别信息，请使用 GetAccountSummary API 操作或 get-account-summary Amazon CLI 命令。

IAM 名称要求

IAM 名称具有以下要求和限制：

- 策略文档只能包含以下 Unicode 字符：水平制表符 (U+0009)、换行符 (U+000A)、回车符 (U+000D) 以及 U+0020 到 U+00FF 范围内的字符。
- 用户、组、角色、策略、实例配置文件、服务器证书和路径的名称必须是字母数字，包括以下常见字符：加号 (+)、等号 (=)、英文逗号 (,)、英文句号 (.)、at 符 (@)、下划线 (_) 和连字符 (-)。路径名称必须以正斜杠 (/) 开始和结束。

- 账户中的用户、组、角色和实例配置文件的名称必须唯一。上述名称不区分大小写，例如，您不能同时创建名为 **ADMINS** 和 **admins** 的组。
- 第三方用于担任角色的外部 ID 值必须至少包含 2 个字符，最多包含 1224 个字符。该值必须是字母数字，没有空格。它还可以包含以下符号：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、@ 符号、冒号 (:)、正斜杠 (/) 和连字符 (-)。有关外部 ID 的更多信息，请参阅 访问第三方拥有的 Amazon Web Services 账户。
- 对于嵌入内联策略的用户、组或角色，这些策略的名称必须唯一。这些名称可以包含任何基本拉丁语（ASCII）字符，但以下保留字符除外：反斜杠（\）、正斜杠（/）、星号（*）、问号（？）和空格。根据 RFC 3986 第 2.2 部分，这些字符是保留字符。
- 用户密码（登录配置文件）可以包含任何基本拉丁语 (ASCII) 字符。
- 所有 Amazon 产品的 Amazon Web Services 账户 ID 别名都必须是唯一的，必须是遵循 DNS 命名惯例的字母数字。别名必须是小写字母，不能以连字符开头或结尾，不能连续使用两个连字符，也不能是 12 个纯数字。

要获取基本拉丁语 (ASCII) 字符列表，请转到 Library of Congress Basic Latin (ASCII) Code Tabl。

IAM 对象配额

配额，也称为 Amazon 中的限制，是 Amazon Web Services 账户 中资源、操作和项目的最大值。使用服务配额管理 IAM 配额。

有关 IAM 服务端点的列表和服务配额，请参阅《Amazon Web Services 一般参考》中的 Amazon Identity and Access Management 端点和限额。

请求提高限额

1. 按照《Amazon 登录用户指南》中的 如何登录 Amazon 所述，根据用户类型选择相应的登录过程，以登录到 Amazon Web Services 管理控制台。
2. 打开服务配额控制台。
3. 在导航窗格中，选择 Amazon 服务。
4. 在导航栏中，选择 美国东部（弗吉尼亚州北部） 区域。然后搜索 **IAM**。
5. 选择 Amazon Identity and Access Management (IAM)，选择配额，然后按照说明请求增加配额。

有关更多信息，请参阅《服务配额用户指南》中的请求增加配额。

要查看有关如何使用服务配额控制台请求增加 IAM 配额的示例，请观看以下视频。

使用服务配额控制台请求增加 IAM 配额。

您可以请求提高可调整 IAM 配额的默认配额。不超过 maximum quota 的请求将自动得到批准，并在几分钟内完成。

下表列举可以自动批准增加限额的资源。

资源	默认配额	最大配额
每个账户中的客户托管式策略数	1500	10000
每个账户的组数	300	500
每个账户的实例配置文件数：	1000	10000
每个角色的托管式策略数	10	25
每个用户的托管式策略数	10	20
每个组的托管式策略数	10	10
角色信任策略长度	2048 个字符	8192 个字符
每个账户的角色数：	1000	10000
每个账户的服务器证书数：	20	20
每个账户的 OpenId Connect 提供商数	100	700

IAM Access Analyzer 配额

有关 IAM Access Analyzer 服务端点的列表和服务配额，请参阅《Amazon Web Services 一般参考》中的 IAM Access Analyzer 端点和限额。

IAM Roles Anywhere 限额

有关 IAM Roles Anywhere 服务端点的列表和服务配额，请参阅《Amazon Web Services 一般参考》中的 Amazon Identity and Access Management Roles Anywhere 端点和限额。

STS 请求配额

Amazon Security Token Service (Amazon STS) 强制执行以下请求配额。

对于使用 Amazon 凭证发出的 Amazon STS 请求，默认请求配额为每账户每区域每秒 600 个请求。以下 Amazon STS 操作共享此配额：

- AssumeRole
- DecodeAuthorizationMessage
- GetAccessKeyInfo
- GetCallerIdentity
- GetFederationToken
- GetSessionToken

> **Note**
>
> Amazon 服务主体向 Amazon STS 发出的请求（例如为与 Amazon 服务结合使用而要用于代入角色的请求）不会消耗账户中每秒 STS 请求的配额。

例如，若一个 Amazon Web Services 账户 在同一区域每秒发出 100 个 GetCallerIdentity 请求和 100 个 AssumeRole 调用，则该账户每秒消耗该区域 600 个可用 STS 请求中的 200 个请求。

对于跨账户 AssumeRole 请求，只有发出 AssumeRole 请求的账户才会影响 STS 配额。目标账户不会消耗任何配额。

要请求增加 STS 的请求配额，请向 Amazon Support 提交工单。

> **Note**
>
> 随着 Amazon STS 全局端点 (`https://sts.amazonaws.com`) 即将发生的更改，对全局端点的请求将不会与默认启用的区域中的 Amazon STS 区域端点共享每秒请求数 (RPS) 配额。当对 Amazon STS STS 全局端点的请求源自单个区域时，它将计入全局端点的 RPS 配额。但是，当请求来自多个区域时，每个额外区域都将获得其自己的独立 RPS 配额。有关 Amazon STS 全局端点更改的更多信息，请参阅 Amazon STS 全局端点更改。

IAM 和 STS 字符限制

以下是 IAM 和 Amazon STS 的最大字符数和大小限制。您不能请求提高以下限制。

说明	限制
Amazon Web Services 账户 ID 的别名	3-63 个字符
对于内联策略	您可以向 IAM 用户、角色或组添加所需数量的内联策略。但是，每个实体的总聚合策略大小（所有内联策略的总大小）不能超过以下限制： • 用户策略大小不得超过 2048 个字符。 • 角色策略大小不得超过 10240 个字符。 • 组策略大小不得超过 5120 个字符。 **Note** 在计算策略大小时，IAM 不会将空格计入这些限制。
用于客户管理型策略	• 每个托管式策略的大小不能超过 6144 个字符。 **Note** 在计算策略大小时，IAM 不会将空格计入这一限制。
组名	128 个字符
实例配置文件名称	128 个字符
登录配置文件的密码	1-128 个字符
路径	512 个字符

说明	限制
策略名称	128 个字符
角色名称	64 个字符 **Important** 如果您通过 Amazon Web Services 管理控制台 中的切换角色功能使用角色，则组合的 `Path` 和 `RoleName` 不能超过 64 个字符。
角色会话持续时间	12 小时 从 Amazon CLI 或 API 中担任角色时，您可以使用 `duration-seconds` CLI 参数或 `DurationSeconds` API 参数请求更长的角色会话。您可以指定 900 秒（15 分钟）到角色的最大会话持续时间设置之间的值，该设置的范围是 1 - 12 小时。如果未指定 `DurationSeconds` 参数值，您的安全凭证的有效期为 1 小时。在控制台内切换角色的 IAM 用户被授予最大会话持续时间或用户会话中的剩余时间（以较少者为准）。最大会话持续时间设置不限制 Amazon 服务建立的会话。要了解如何查看您的角色的最大值，请参阅更新角色的最长会话持续时间。
角色会话名称	64 个字符

说明	限制
角色会话策略	• 传递的 JSON 策略文档和所有传递的托管式策略 ARN 字符合计大小不能超过 2048 个字符。 • 创建会话时，最多可传递 10 个托管策略 ARN。 • 在以编程方式为角色或 Amazon STS 联合用户主体创建临时会话时，您只能传递一个 JSON 策略文档。 • 此外，Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制文件格式。`PackedPolicySize` 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。 • 建议您使用 Amazon CLI 或 Amazon API 传递会话策略。Amazon Web Services 管理控制台可能会将其他控制台会话信息添加到打包策略中。
角色会话标签	• 会话标签必须满足 128 个字符的标签键限制和 256 个字符的标签值限制。 • 您最多可以传递 50 个会话标签。 • Amazon 转换会将传递的会话策略和会话标签压缩为具有单独限制的打包二进制格式。您可以使用 Amazon CLI 或 Amazon API 传递会话标签。`PackedPolicySize` 响应元素指示您请求的策略和标签接近大小上限的程度，以百分比来表示。
经过 base64 编码的 SAML 身份验证	100000 个字符 此字符限制适用于 `assume-role-with-saml` CLI 或 `AssumeRoleWithSAML` API 操作。

说明	限制
标记密钥	128 个字符 此字符限制适用于 IAM 资源的标签和会话标签。
标记值	256 个字符 此字符限制适用于 IAM 资源的标签和会话标签。 标签值可以为空，这意味着标签值的长度可以为 0 个字符。
IAM 创建的唯一 ID	128 个字符。例如： • 以 AIDA 开头的用户 ID • 以 AGPA 开头的组 ID • 以 AROA 开头的角色 ID • 以 ANPA 开头的托管策略 ID • 以 ASCA 开头的服务器证书 ID Note 这不是一个详尽的列表，也不保证某种类型的 ID 仅以指定的字母组合开始。
用户名称	64 个字符

双堆栈端点支持

双堆栈端点让客户端能够使用 IPv4 或 IPv6 地址与 Amazon 服务通信。IAM 和 Amazon STS 均支持双堆栈端点。有关如何将 VPC 配置为双堆栈模式的更多信息，请参阅《Amazon VPC 用户指南》中的对 VPC 的 IPv6 支持。

如果您使用的 IAM 策略包含 aws:sourceIp 或 aws:vpcSourceIp 条件键，则需要更新这些策略以支持 IPv6 客户端。有关为 VPC 分配 IPv6 地址的更多信息，请参阅《Amazon VPC 用户指南》中的为 VPC 和子网分配 IP 地址。

IAM 双堆栈端点支持

IAM 提供双堆栈公有端点，支持 IPv4 和 IPv6 客户端。IAM 双堆栈公有端点也可以使用 Amazon PrivateLink 从虚拟私有云 (VPC) 进行私密访问。有关为 IAM 创建私有接口 VPC 端点的更多信息，请参阅为 IAM 创建 VPC 端点。

IAM 双堆栈端点是 https://iam.global.api.amazonwebservices.com.cn。

与双堆栈公有端点不同的是，https://iam.cn-north-1.amazonaws.com.cn 的 IAM 公有端点仅支持 IPv4 客户端。当使用 Amazon PrivateLink 对 VPC 进行私密访问时，IAM 公有端点可同时支持 IPv4 和 IPv6 客户端。

Amazon STS 双堆栈端点支持

Amazon STS 提供同时支持 IPv4 和 IPv6 客户端的双堆栈区域端点。Amazon STS 双堆栈区域端点也可以使用 Amazon PrivateLink 从虚拟私有云（VPC）进行私密访问。有关为 Amazon STS 创建私有接口 VPC 端点的更多信息，请参阅为 Amazon STS 创建 VPC 端点。

下表按分区显示 Amazon STS 双堆栈区域端点：

分区	双堆栈端点 URL
商用 Amazon 区域	常规端点：https://sts. *<region>*.api.aws FIPS 端点：https://sts-fips. *<region>*.api.aws
Amazon GovCloud (US) 区域	常规端点：https://sts. *<region>*.api.aws
中国区域	常规端点：https://sts. *<region>*.api.amaz onwebservices.com.cn

> **Note**
>
> FIPS 端点仅适用于美国和加拿大区域。

限制

以下限制适用于 Amazon STS 双堆栈端点支持：

- 全局端点 `https://sts.amazonaws.com` 不支持 IPv6 客户端。必须使用双堆栈区域端点才能支持 IPv6 客户端。
- 不支持仅 IPv6 Amazon STS VPC 端点。可以将 VPC 端点配置为 IPv4 或双堆栈连接。

接口 VPC 端点

如果您使用 Amazon Virtual Private Cloud（Amazon VPC）托管 Amazon 资源，则可以在您的 VPC 和 Amazon Identity and Access Management（IAM）或 Amazon Security Token Service（Amazon STS）之间建立私有连接。您可以使用此连接实现 IAM 或 Amazon STS 与您的 VPC 中资源的通信而不用访问公共互联网。

Amazon VPC 是一项 Amazon 服务，可用来启动在虚拟网络中定义的 Amazon 资源。借助 VPC，您可以控制您的网络设置，如 IP 地址范围、子网、路由表和网络网关。要将 VPC 连接到 IAM 或 Amazon STS，请为每个服务定义一个接口 VPC 端点。该端点提供了到 IAM 或 Amazon STS 的可靠、可扩展的连接，无需互联网网关、网络地址转换 (NAT) 实例或 VPN 连接。有关更多信息，请参阅《Amazon VPC 用户指南》中的什么是 Amazon VPC？。

接口 VPC 端点由 Amazon PrivateLink 提供支持，后者是一种 Amazon 技术，可将弹性网络接口与私有 IP 地址结合使用来支持 Amazon 服务之间的专有通信。有关更多信息，请参阅 Amazon 服务 的 Amazon PrivateLink。

以下信息面向 Amazon VPC 的用户。有关更多信息，请参阅《Amazon VPC 用户指南》中的 Amazon VPC 入门。

主题

- VPC 端点可用性
- 为 IAM 创建 VPC 端点
- 为 Amazon STS 创建 VPC 端点

VPC 端点可用性

> **Important**
>
> 由只能在 IAM 控制面板所在的区域中为 IAM 创建接口 VPC 端点。如果您的 VPC 与 IAM 控制面板区域位于不同的区域，则必须使用 Amazon Transit Gateway 以允许从另一个区域访问 IAM 接口 VPC 端点。有关更多信息，请参阅 为 IAM 创建 VPC 端点。

IAM 当前在以下区域中支持 VPC 端点：

- 美国东部（弗吉尼亚州北部）
- 中国（北京）
- Amazon GovCloud（美国西部）

Amazon STS 当前在以下区域中支持 VPC 端点：

- 美国东部（弗吉尼亚州北部）
- 美国东部（俄亥俄州）
- 美国西部（加利福尼亚北部）
- 美国西部（俄勒冈州）
- 非洲（开普敦）
- 亚太地区（香港）
- 亚太地区（海得拉巴）
- 亚太地区（雅加达）
- 亚太地区（墨尔本）
- 亚太地区（孟买）
- 亚太地区（大阪）
- 亚太地区（首尔）
- 亚太地区（新加坡）
- 亚太地区（悉尼）
- 亚太地区（东京）
- 加拿大（中部）

- 加拿大西部（卡尔加里）
- 中国（北京）
- 中国（宁夏）
- 欧洲地区（法兰克福）
- 欧洲地区（爱尔兰）
- 欧洲地区（伦敦）
- 欧洲地区（米兰）
- 欧洲地区（巴黎）
- 欧洲（西班牙）
- 欧洲地区（斯德哥尔摩）
- 欧洲（苏黎世）
- 以色列（特拉维夫）
- 中东（巴林）
- 中东（阿联酋）
- 南美洲（圣保罗）
- Amazon GovCloud（美国东部）
- Amazon GovCloud（美国西部）

为 IAM 创建 VPC 端点

要开始将 IAM 与您的 VPC 一起使用，请为 IAM 创建接口 VPC 端点。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用接口 VPC 端点访问 Amazon 服务。

由只能在 IAM 控制面板所在的区域中为 IAM 创建接口 VPC 端点。在商业 Amazon Web Services 区域 中，IAM 控制面板位于美国东部（弗吉尼亚州北部）区域（us-east-1）。适用于 IAM 的 Amazon PrivateLink 接口 VPC 端点服务名称为 `com.amazonaws.iam`。有关支持 IAM 的 VPC 端点的 Amazon Web Services 区域 列表，请参阅 VPC 端点可用性。

如果您的 VPC 与 IAM 控制面板区域位于不同的区域，则必须使用 Amazon Transit Gateway 以允许从另一个区域访问 IAM 接口 VPC 端点。

使用 Amazon Transit Gateway 从其他区域的 VPC 访问 IAM 接口 VPC 端点

1. 创建中转网关，或者使用现有的中转网关互连您的虚拟私有云（VPC）。每个区域都需要中转网关。有关更多信息，请参阅《Amazon Transit Gateway 指南》中的创建中转网关。

2. 创建中转网关 VPC 连接以将每个 VPC 连接到中转网关。有关更多信息，请参阅《Amazon Transit Gateway 指南》中的创建中转网关连接。
3. 创建中转网关 VPC 对等连接以在对等 VPC 之间路由流量。有关更多信息，请参阅《Amazon Transit Gateway 指南》中的创建对等连接。

> **Note**
>
> VPC 对等连接也可以在对等 VPC 之间路由流量，但是在有大量 VPC 时这种方法不能很好地扩展。我们建议不要使用 VPC 对等连接，而是使用 Amazon Transit Gateway 对等连接，通过可扩展的中央枢纽改进 VPC 和本地网络管理。有关 VPC 对等连接的更多信息，请参阅《Amazon VPC 对等指南》中的使用 VPC 对等连接。

为 Amazon STS 创建 VPC 端点

要开始将您的 Amazon STS 与 VPC 一起使用，请为 Amazon STS 创建接口 VPC 端点。有关更多信息，请参阅《Amazon VPC 用户指南》中的使用接口 VPC 端点访问 Amazon 服务。

在创建 VPC 端点后，您必须使用匹配的区域端点发送 Amazon STS 请求。Amazon STS 建议您同时使用 `setRegion` 和 `setEndpoint` 方法以对区域端点进行调用。您可以仅将 `setRegion` 方法用于手动启用的区域，例如，亚太区域（香港）。在这种情况下，调用将定向到 STS 区域端点。要了解如何手动启用区域，请参阅《Amazon Web Services 一般参考》中的 管理 Amazon 区域。如果仅将 `setRegion` 方法用于默认启用的区域，调用将定向到全球端点 `https://sts.amazonaws.com` 。

在您使用区域端点时，Amazon STS 使用公有端点或私有接口 VPC 端点（二者中正在使用的那个）调用其他 Amazon 服务。例如，假设您已为 Amazon STS 创建了接口 VPC 端点，并且已经从位于 VPC 中的资源的 Amazon STS 请求了临时凭证。在这种情况下，这些凭证默认开始流经接口 VPC 端点。有关使用 Amazon STS 发出区域请求的更多信息，请参阅 管理 Amazon Web Services 区域中的 Amazon STS。

使用 IAM 的 Amazon 服务

下面列出的 Amazon 服务根据字母顺序分组，包含所支持的 IAM 功能的信息：

- 服务 - 您可以选择一项服务的名称，以查看有关该服务的 IAM 授权和访问权限的 Amazon 文档。

- 操作 – 您可以指定策略中的各项操作。如果服务不支持此功能，则 visual editor（可视化编辑器）中将选中 All actions（所有操作）。在 JSON 策略文档中，您必须在 * 元素中使用 Action。有关每个服务中的操作列表，请参阅 Amazon 服务的操作、资源和条件键。
- 资源级权限 – 您可以使用 ARN 在策略中指定各个资源。如果服务不支持此功能，则 policy visual editor（策略可视化编辑器）中将选中 All resources（所有资源）。在 JSON 策略文档中，您必须在 * 元素中使用 Resource。某些操作 (如 List* 操作) 不支持指定 ARN，因为它们被设计为返回多个资源。如果某项服务只针对部分资源支持此功能，将在表格中用 Partial（部分）标明。有关更多信息，请参阅该服务的文档。
- 基于资源的策略 – 您可以将基于资源的策略附加到服务中的某项资源。基于资源的策略包含 Principal 元素，用于指定可以访问此资源的 IAM 身份。有关更多信息，请参阅 基于身份的策略和基于资源的策略。
- ABAC（基于标签的授权） – 要利用标签来控制访问权限，您需要在策略的 条件元素 中使用 aws:ResourceTag/*key-name*、aws:RequestTag/*key-name* 或 aws:TagKeys 条件键提供标签信息。如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为 Partial（部分）。有关基于标签等属性定义权限的更多信息，请参阅 使用 ABAC 授权根据属性定义权限。要查看有关 ABAC 设置步骤的教程，请参阅使用基于属性的访问权限控制 (ABAC)。
- Temporary credentials（临时凭证）- 您可以使用在通过 IAM Identity Center 登录时获得的短期凭证、在控制台中切换角色或在 Amazon CLI 或 Amazon API 中使用 Amazon STS 生成的临时凭证。只有在使用长期 IAM 用户凭证时，您才能访问带有否值的服务。这包括用户名和密码或用户访问密钥。有关更多信息，请参阅 IAM 临时安全凭证。
- 服务相关角色 – 服务相关角色是一种特殊类型的服务角色，可授予服务代表您访问其他服务中的资源的权限。选择 是或部分链接可查看文档，了解支持这些角色的服务。此列不指示服务是否使用标准服务角色。有关更多信息，请参阅服务相关角色。
- 更多信息 – 如果服务不能完全支持某项功能，您可以检查该条目的脚注，查看限制以及相关信息的链接。

使用 IAM 的服务

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon 账户管理	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 操作建议	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Activate Console	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon AI 操作	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Amplify 管理员	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Amplify	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✗ 不支持
Amazon Amplify UI Builder	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
适用于 Amazon MSK 集群的 Apache Kafka API	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon API Gateway	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是
Amazon API Gateway 管理	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon API Gateway 管理第 2 版	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon App Studio	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon App2Container	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon AppConfig	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon AppFabric	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon AppFlow	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon AppIntegrations	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Application Auto Scaling	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Application Discovery Arsenal	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Application Discovery Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Transform MGN	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Application Recovery Controller - 可用区转移	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Application Transformation Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon App Mesh	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon App Mesh 预览	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon App Runner	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon AppStream 2.0	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon AppSync	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon ARC 区域切换	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Artifact	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Athena	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Audit Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Aurora DSQL	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Auto Scaling	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon B2B Data Interchange	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Backup	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Backup 网关	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Backup 搜索	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Backup 存储	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Batch	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Bedrock	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Bedrock Agentcore	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✗ 不支持
Amazon 账单与成本管理	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 账单与成本管理控制面板	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 账单与成本管理 Data Exports	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 账单与成本管理 定价计算器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 账单与成本管理 建议的操作	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Billing Conductor	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Braket	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Budget 服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Certificate Manager (ACM)	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
聊天应用程序中的 Amazon Q 开发者版	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Chime	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Clean Rooms	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Clean Rooms ML	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Client VPN	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Cloud9	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Web Services 云 Control API	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Cloud Directory	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon CloudFormation	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudFront	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✓ 是
Amazon CloudFront KeyValueStore	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudHSM	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Cloud Map	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudSearch	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CloudShell	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CloudTrail	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	✓ 是	✓ 是
Amazon CloudTrail 数据	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分（信息）
Amazon CloudWatch Application Insights	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon CloudWatch Application Signals	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch Evidently	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch 网络监测仪	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch Logs	✓ 是	✓ 是	✓ 是	⚠ 部分	✓ 是	✓ 是
Amazon CloudWatch 网络综合监测仪	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch Observability Access Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch 可观测性管理服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CloudWatch RUM	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon CloudWatch Synthetics	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeArtifact	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon CodeBuild	✓ 是	✓ 是	✓ 是（信息）	⚠ 部分（信息）	✓ 是	✗ 不支持
Amazon CodeCatalyst	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon CodeCommit	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeConnections	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeDeploy	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodeDeploy 安全主机命令服务	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon CodeGuru Profiler	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon CodeGuru Reviewer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon CodeGuru 安全防御工具	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon CodePipeline	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS CodeStar	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS CodeStar 连接	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
AWS CodeStar 通知	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon CodeWhisperer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Cognito	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Cognito Sync	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Cognito 用户群体	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Comprehend	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Comprehend Medical	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Compute Optimizer	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Config	✓ 是	⚠ 部分（信息）	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Connect Customer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Connect Customer Cases	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Connect Customer Profiles	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Connect Customer Outbound Campaigns	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Connect Customer Voice ID	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Connect Health	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✗ 不支持
Amazon Console Mobile Application	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon整合账单	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 控制目录	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Control Tower	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 成本和使用情况报告	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Cost Explorer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 成本优化中心	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 客户验证服务	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Database Migration Service	✓ 是	✓ 是	✗ 否（信息）	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Database Query Metadata Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Data Exchange	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Data Lifecycle Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Data Pipeline	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✗ 不支持
Amazon DataSync	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon DataZone	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Deadline Cloud	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon DeepComposer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon DeepRacer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Detective	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Device Farm	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon DevOps Guru	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon 诊断工具	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Direct Connect	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Directory Service	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 目录服务数据	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon DocumentDB Elastic Clusters	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon DynamoDB Accelerator (DAX)	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon DynamoDB	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Elastic Compute Cloud (Amazon EC2)	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	⚠ 部分（信息）
Amazon EC2 Auto Scaling	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
EC2 Image Builder	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon EC2 Instance Connect	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Elastic Beanstalk	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Elastic Block Store（Amazon EBS）	✓ 是	⚠ 部分	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Elastic Container Registry（Amazon ECR）	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Elastic Container Registry 公有（Amazon ECR 公有）	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Elastic Container Service（Amazon ECS）	✓ 是	⚠ 部分（信息）	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 弹性灾难恢复	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Elastic File System (Amazon EFS)	✓ 是	✓ 是	✓ 是	⚠ 部分	✓ 是	✓ 是
Amazon Elastic Kubernetes Service(Amazon EKS)	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Elastic Kubernetes Service（Amazon EKS）Auth	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Elastic Load Balancing	✓ 是	⚠ 部分	✗ 不支持	⚠ 部分	✓ 是	✓ 是
Amazon Elastic Transcoder	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Elastic VMware Service	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon ElastiCache	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Elemental Appliances and Software 激活服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Elemental Appliances and Software	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS Elemental MediaConnect	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
AWS Elemental MediaConvert	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS Elemental MediaLive	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS Elemental MediaPackage	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分（信息）
AWS Elemental MediaPackageV2	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
AWS Elemental MediaPackage VOD	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分（信息）
AWS Elemental MediaStore	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
AWS Elemental MediaTailor	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Elemental Support 案例	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Elemental Support 内容	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon EMR	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon EMR on EKS	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon EMR Serverless	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon End User Messaging SMS and Voice V2	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 最终用户消息社交	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Entity Resolution 数据匹配服务	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon EventBridge	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon EventBridge Pipes	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon EventBridge 调度器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon EventBridge Schemas	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon 发票管理	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Fault Injection Service	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon FinSpace	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon FinSpace API	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Firewall Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分
Fleet Hub for Amazon IoT Device Management	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Forecast	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Fraud Detector	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
FreeRTOS	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 免费套餐	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon FSx	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon GameLift Servers	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon GameLift Streams	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Global Accelerator	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Glue	✓ 是	✓ 是	⚠ 部分	⚠ 部分	✓ 是	✗ 不支持
Amazon Glue DataBrew	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Ground Station	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Ground Truth Labeling	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon GuardDuty	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Health API 和通知	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon HealthImaging	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon HealthLake	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon HealthOmics	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IAM Identity Center	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✓ 是
IAM Identity Center 目录	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
IAM Identity Center Identity Store	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
IAM Identity Center OIDC 服务	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Identity and Access Management（IAM）	✓ 是	✓ 是	⚠ 部分（信息）	⚠ 部分（信息）	⚠ 部分（信息）	✗ 不支持
Amazon Identity and Access Management and Access Analyzer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分
Amazon Identity and Access Management Roles Anywhere	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Identity Store Auth	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Identity Sync	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Import/Export	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Inspector	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Inspector Classic	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon InspectorScan	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Interactive Video Service	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Interactive Video Service Chat	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Web Services 开票	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT Analytics	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	✓ 是	✗ 不支持
Amazon IoT Core Device Advisor	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT Device Tester	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon IoT Events	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT FleetWise	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT Greengrass	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IoT GreengrassV2	✓ 是	✓ 是	✗ 不支持	⚠ 部分	✓ 是	✗ 不支持
Amazon IoT Jobs DataPlane	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon IoT 托管集成服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon IoT SiteWise	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon IoT TwinMaker	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon IoT Wireless	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon IQ	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon IQ 权限	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Kendra	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Kendra Intelligent Ranking	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Key Management Service (Amazon KMS)	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Keyspaces（Apache Cassandra 兼容）	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
适用于 Apache Flink 的亚马逊托管服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
适用于 Apache Flink 的亚马逊托管服务 V2	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Data Firehose	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Kinesis Data Streams	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon Kinesis Video Streams	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Lake Formation	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Lambda	✓ 是	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	⚠ 部分（信息）
Amazon Launch Wizard	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Lex	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Lex V2	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是
Amazon License Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon License Manager Linux Subscriptions Manager	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon License Manager 用户订阅	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Lightsail	✓ 是	⚠ 部分（信息）	✗ 不支持	⚠ 部分（信息）	✓ 是	✓ 是
Amazon Location Service	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Location Service 地图	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Location Service 位置	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Location Service 路线	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Lookout for Equipment	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Lookout for Metrics	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Lookout for Vision	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Machine Learning	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Macie	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Mainframe Modernization	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Mainframe Modernization Application Testing	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Managed Blockchain	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Managed Blockchain 查询	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Managed Grafana	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Managed Service for Prometheus	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Managed Streaming for Apache Kafka (MSK)	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	✓ 是	✓ 是
Amazon Managed Streaming for Kafka Connect	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Managed Workflows for Apache Airflow	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Web Services Marketplace	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Web Services Marketplace 目录	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Marketplace Commerce Analytics	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Web Services Marketplace 部署服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Web Services Marketplace 发现	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Marketplace Entitlement Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace Image Building Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace 管理门户	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Marketplace Metering Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace Private Marketplace	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace Procurement Systems Integration	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace 报告	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace 卖家报告	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services Marketplace Vendor Insights	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Mechanical Turk	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon MediaImport	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持
Amazon MemoryDB	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Message Delivery Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Message Gateway Service	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Microservice Extractor for .NET	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon迁移加速计划服务抵扣金额	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Migration Hub	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Migration Hub Orchestrator	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Migration Hub Refactor Spaces	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Migration Hub Strategy Recommendations	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Monitron	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon MQ	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Neptune	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
多方审批	✓ 是	✓ 是	✗ 不支持	⚠ 部分（信息）	✓ 是	✗ 不支持
Amazon Neptune Analytics	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Network Firewall	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
网络流量监测仪	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Network Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是（信息）
Amazon Network Manager Chat	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Nimble Studio	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon One Enterprise	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon OpenSearch	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon OpenSearch Ingestion	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon OpenSearch Serverless	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon OpenSearch Service	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Oracle Database@Amazon	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon OpsWorks	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon OpsWorks 配置管理	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Organizations	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是
Amazon Outposts	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Panorama	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 并行计算服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 合作伙伴中央账户管理	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 合作伙伴中心销售	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Payment Cryptography	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Payments	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 性能详情	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Personalize	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Pinpoint	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Pinpoint 电子邮件服务	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Pinpoint 短信和语音服务	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Polly	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 价目表	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 私有 5G	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
适用于 Active Directory 的 Amazon 私有 CA 连接器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 私有 CA Connector for SCEP	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon 私有证书颁发机构 (Amazon 私有 CA)	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon PrivateLink	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Proton	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 采购订单控制台	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Q Business	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Q Business Q 应用	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Q 开发者版	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Q in Connect	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Quantum Ledger Database (Amazon QLDB)	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Quick	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon RDS Data API	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon RDS IAM 身份验证	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon 回收站	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Redshift	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Redshift Data API	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Redshift Serverless	✓ 是	✓ 是	✓ 是（信息）	✓ 是	✓ 是	✗ 不支持
Amazon Rekognition	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	✓ 是	✗ 不支持
Amazon Relational Database Service（Amazon RDS）（信息）	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Web Services re:Post Private	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Resilience Hub	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Resource Access Manager (Amazon RAM)	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 资源探索器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Resource Groups	✓ 是	✓ 是	✗ 不支持	✓ 是	⚠ 部分（信息）	✓ 是
Amazon Resource Groups Tagging API	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon RHEL 知识库门户	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon RoboMaker	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Route 53	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Route 53 域	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持
Amazon Route 53 Profiles	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Route 53 Recovery 集群	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Route 53 恢复控制配置	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Route 53 Recovery 就绪性	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Route 53 Resolver	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon RTB Fabric	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon S3 Express	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon Glacier	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	⚠ 部分
Amazon S3 表	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	✗ 不支持
Amazon S3 Vectors	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon SageMaker AI	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	⚠ 部分（信息）
Amazon SageMaker AI 数据科学助手	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon SageMaker AI 地理空间功能	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
带有 MLflow 的 Amazon SageMaker AI	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Savings Plans	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Secrets Manager	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon Security Hub CSPM	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 安全事件响应	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Security Lake	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Security Token Service (Amazon STS)	✓ 是	⚠ 部分（信息）	✗ 不支持	✓ 是	⚠ 部分（信息）	✗ 不支持
Amazon Serverless Application Repository	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	✗ 不支持
Amazon Service Catalog	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
用于管理 Amazon 控制台用户体验功能的 Amazon 服务	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
服务配额	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Shield	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Shield 网络安全分析器	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Signer	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 登录	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon SimpleDB	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Simple Email Service - Mail Manager	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Simple Email Service (Amazon SES) 第 2 版	✓ 是	⚠ 部分（信息）	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是
Amazon Simple Notification Service (Amazon SNS)	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✗ 不支持
Amazon Simple Queue Service (Amazon SQS)	✓ 是	✓ 是	✓ 是	✓ 部分	✓ 是	✗ 不支持
Amazon Simple Storage Service（Amazon S3）	✓ 是	✓ 是	✓ 是	⚠ 部分（信息）	✓ 是	⚠ 部分（信息）
Amazon Simple Storage Service (Amazon S3) 对象 Lambda	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Outposts 上的 Amazon Simple Storage Service (Amazon S3)	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是
Amazon Simple Workflow Service (Amazon SWF)	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon SimSpace Weaver	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Site-to-Site VPN	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Snowball Edge	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Snowball Edge 边缘	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Snowball Edge Device Management	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon SQL Workbench	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Step Functions	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Storage Gateway	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Supply Chain	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Web Services 支持	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✓ 是
Amazon Web Services Support App in Slack	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services 支持 控制台	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Web Services 支持 计划	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Sustainability	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Systems Manager	✓ 是	✓ 是	⚠ 部分	✓ 是	✓ 是	✓ 是
Amazon Systems Manager for SAP	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Systems Manager GUI Connect	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Systems Manager Incident Manager	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是	✓ 是
Amazon Systems Manager Incident Manager Contacts	✓ 是	✓ 是	✓ 是	✗ 不支持	✓ 是	✗ 不支持
Amazon Systems Manager 快速设置	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
标签编辑器	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 税务设置	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon 电信网络生成器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Textract	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Timestream	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Timestream Influxdb	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Tiros API（用于 Reachability Analyzer）	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持	✗ 不支持
Amazon Transcribe	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Transfer Family	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 转换	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Translate	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Trusted Advisor	⚠ 部分（信息）	✓ 是	✗ 不支持	✗ 不支持	⚠ 部分	✓ 是
Amazon 用户通知	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon 用户通知联系人	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon 用户订阅	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Verified Access	✓ 是	✗ 不支持	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon Verified Permissions	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon Virtual Private Cloud (Amazon VPC)	✓ 是	⚠ 部分（信息）	⚠ 部分（信息）	✓ 是	✓ 是	⚠ 部分（信息）
Amazon VPC Lattice	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon VPC Lattice Service	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon WAF	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon WAF Classic	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon WAF 区域性	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon Well-Architected Tool	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon Wickr	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon WorkMail	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon WorkMail Message Flow	✓ 是	✓ 是	✗ 不支持	✗ 不支持	✓ 是	✗ 不支持
Amazon WorkSpaces	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持

服务	操作	资源级权限	基于资源的策略	ABAC	临时凭证	服务关联角色
Amazon WorkSpaces 托管型实例	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon WorkSpaces 安全浏览器	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✓ 是
Amazon WorkSpaces 瘦客户端	✓ 是	✓ 是	✗ 不支持	✓ 是	✓ 是	✗ 不支持
Amazon X-Ray	✓ 是	⚠ 部分（信息）	✗ 不支持	⚠ 部分（信息）	✓ 是	✗ 不支持

更多信息

Amazon CloudTrail

CloudTrail 在用于将 CloudTrail Lake 与 Amazon 外部的事件源集成的 CloudTrail Lake 事件数据存储、控制面板和通道上支持基于资源的策略。

Amazon CloudWatch

无法使用 Amazon Web Services 管理控制台 创建 CloudWatch 服务相关角色，这些角色仅支持警报操作功能。

Amazon CodeBuild

CodeBuild 支持使用 Amazon RAM 进行跨账户资源共享。

CodeBuild 支持将 ABAC 用于基于项目的操作。

Amazon Config

对于多账户多区域数据聚合和 Amazon Config 规则，Amazon Config 支持资源级权限。有关受支持的资源列表，请参阅 Amazon Config API Guide 中的 Multi-Account Multi-Region Data Aggregation 部分与 Amazon Config Rules 部分。

Amazon Database Migration Service

您可以创建和修改附加到您创建的 Amazon KMS 加密密钥的策略，以加密迁移到支持的目标端点的数据。支持的目标端点包括 Amazon Redshift 和 Amazon S3。有关更多信息，请参阅《Amazon Database Migration Service 用户指南》中的 创建和使用 Amazon KMS 密钥以加密 Amazon Redshift 目标数据 和 创建 Amazon KMS 密钥以加密 Amazon S3 目标对象。

Amazon Elastic Compute Cloud

Amazon EC2 仅针对以下功能支持服务相关角色：竞价型实例请求、竞价型实例集请求、Amazon EC2 Fleet 和 Windows 实例的快速启动。

Amazon Elastic Container Service

仅某些 Amazon ECS 操作支持资源级权限。

AWS Elemental MediaPackage

MediaPackage 支持将客户访问日志发布到 CloudWatch 的服务相关角色，但不支持其他 API 操作。

Amazon Identity and Access Management

IAM 仅支持一种类型的基于资源的策略（称为角色信任策略），这种策略附加到 IAM 角色。有关更多信息，请参阅 向用户授予切换角色的权限。

IAM 为大多数 IAM 资源支持基于标签的访问控制。有关更多信息，请参阅 Amazon Identity and Access Management 资源的标签。

只能使用临时凭证调用 IAM 的一部分 API 操作。有关更多信息，请参阅比较您的 API 选项。

Amazon IoT

连接到 Amazon IoT 的设备通过 X.509 证书或 Amazon Cognito 身份进行身份验证。您可以将 Amazon IoT 策略附加到 X.509 证书或 Amazon Cognito 身份以控制设备有权执行哪些操作。有关更多信息，请参阅《Amazon IoT 开发人员指南》中的 Amazon IoT 的安全和身份。

Amazon Lambda

Lambda 对函数、事件源映射和代码签名配置支持基于属性的访问权限控制 (ABAC)。不支持层。有关更多信息，请参阅在 Lambda 中使用基于属性的访问控制。

Lambda 不具有服务相关角色，但 Lambda@Edge 具有。有关更多信息，请参阅《Amazon CloudFront 开发人员指南》中的 Lambda@Edge 的服务相关角色。

Amazon Lightsail

Lightsail 部分支持资源级权限和 ABAC。有关更多信息，请参阅 Amazon Lightsail的操作、资源和条件键。

Amazon Managed Streaming for Apache Kafka (MSK)

您可以将集群策略附加到已配置为支持多 VPC 连接的 Amazon MSK 集群。

多方审批

对于多方审批身份来源和审批团队，您可以使用所有三个 ABAC 条件键：`aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 和 `aws:TagKeys`。

对于会话，您只能使用 `aws:ResourceTag/key-name` 条件键，因为：

- 会话不是可直接标记的资源。
- 会话从与其关联的审批团队继承标签。

Amazon Network Manager

Amazon Cloud WAN 还支持服务相关角色。有关更多信息，请参阅《Amazon VPC Amazon Cloud WAN 指南》中的 Amazon Cloud WAN 相关角色。

Amazon Redshift Serverless

Redshift Serverless 支持基于资源的策略来共享快照。

Amazon Relational Database Service

Amazon Aurora 是一个与 MySQL 和 PostgreSQL 兼容的完全托管式的关系数据库引擎。在通过 Amazon RDS 设置新的数据库服务器时，您可以选择 Aurora MySQL 或 Aurora PostgreSQL 作为数据

库引擎选项。有关更多信息，请参阅《Amazon Aurora 用户指南》中的 Amazon Aurora 的身份和访问管理。

Amazon Rekognition

基于资源的策略仅支持复制 Amazon Rekognition Custom Labels 模型。

Amazon Resource Groups

用户可以通过允许资源组操作的策略担任角色。

Amazon SageMaker AI

服务相关角色目前可用于 SageMaker AI Studio 和 SageMaker AI 训练作业。

Amazon Security Token Service

Amazon STS 没有“资源”，但允许以与用户相似的方式限制访问。有关更多信息，请参阅根据名称拒绝访问临时安全凭证。

仅 Amazon STS 的一部分 API 操作支持使用临时凭证进行调用。有关更多信息，请参阅比较您的 API 选项。

Amazon Simple Email Service

您只能在引用与发送电子邮件相关的操作（如 `ses:SendEmail` 或 `ses:SendRawEmail`）的策略语句中使用资源级权限。对于引用任何其他操作的策略语句，Resource 元素只能包含 *。

仅 Amazon SES API 支持临时安全凭证。Amazon SES SMTP 接口不支持从临时安全凭证派生的 SMTP 凭证。

Amazon Simple Storage Service

Amazon S3 支持对目录存储桶、对象资源、Amazon S3 Storage Lens 存储统计管理工具、Storage Lens 组和接入点进行基于标签的授权。

Amazon S3 支持针对 Amazon S3 Storage Lens 存储统计管理工具的服务相关角色。

Amazon Trusted Advisor

针对 Trusted Advisor 的 API 访问通过 Amazon Web Services 支持 API 执行，并受 Amazon Web Services 支持 IAM 策略的控制。

Amazon Virtual Private Cloud

Amazon VPC 支持通过将单个资源策略附加到 VPC 端点来限制可通过该端点访问的内容。有关使用基于资源的策略控制特定 Amazon VPC 端点访问资源的更多信息，请参阅《Amazon PrivateLink 用户指南》中的 使用端点策略控制服务访问。

Amazon X-Ray

X-Ray 并非对所有操作均支持资源级权限。

X-Ray 支持针对组和采样规则实现基于标签的访问控制。

适用于 API 请求的 Amazon 签名版本 4

> **Important**
>
> 如果您使用 Amazon SDK（请参阅示例代码和库）或 Amazon Command Line Interface（Amazon CLI）工具向 Amazon 发送 API 请求，则可以跳过签名过程，因为 SDK 和 CLI 客户端会使用您提供的访问密钥来验证您的请求。除非您有充分的理由不这样做，否则我们建议您始终使用 SDK 或 CLI。
>
> 在支持多个签名版本的区域中，手动签名请求意味着您必须指定要使用的签名版本。当您向多区域访问点提供请求时，SDK 和 CLI 会自动切换为使用签名版本 4A，而无需进行其他配置。

您在请求中发送的身份验证信息必须包含签名。Amazon签名版本 4（SigV4）是将身份验证信息添加到 Amazon API 请求的 Amazon 签名协议。

您无法使用秘密访问密钥对 API 请求进行签名。相反，您可以使用 Sigv4 签名过程。签名请求涉及：

1. 根据请求详细信息创建规范请求。
2. 使用 Amazon 凭证计算签名。
3. 将此签名作为授权标头添加到请求。

然后 Amazon 复制此过程并验证签名，相应地授予或拒绝访问权限。

Symmetric SigV4 要求您在特定日期派生一个密钥，该密钥的作用域限定为单个 Amazon 区域的单个 Amazon 服务。这让各区域的密钥和计算得出的签名均会有所不同，也就是说，您必须知道签名要发送到哪个区域。

非对称签名版本 4（SigV4a）是支持使用新算法进行签名的扩展版本，可以生成可在多个 Amazon 区域验证的单个签名。借助 SigV4a，您可以签署多个区域的请求，在多个区域之间实现无缝路由和失效转移。在使用 Amazon SDK 或 Amazon CLI 来调用需要多区域签名的功能时，签名类型会自动改为使用 SigV4a。有关详细信息，请参阅 Amazon SigV4a 的工作原理。

Amazon SigV4 工作原理

以下步骤介绍了使用 SigV4 计算签名的一般过程：

1. 待签字符串取决于请求类型。例如，使用 HTTP 授权标头或查询参数进行身份验证时，可以使用请求元素组合来创建待签字符串。对于 HTTP POST 请求，请求中的 POST 策略是您签名的字符串。
2. 签名密钥是一系列计算，每个步骤的结果都输入到下一个步骤中。最后一步是签名密钥。
3. Amazon 服务收到经身份验证请求时，其会使用请求中包含的身份验证信息重新创建签名。如果签名匹配，则服务将处理该请求。否则，服务将拒绝该请求。

有关更多信息，请参阅 Amazon API 请求签名的元素。

Amazon SigV4a 的工作原理

SigV4a 使用基于公私密钥加密的非对称签名。SigV4a 的作用域凭证派生过程与 SigV4 类似，唯一差异在于 SigV4a 使用相同密钥对所有请求进行签名，而无需根据日期、服务和区域派生不同的签名密钥。椭圆曲线数字签名算法（ECDSA）密钥对可由现有的 Amazon 秘密访问密钥派生而来。

此系统使用非对称加密来验证多区域签名，因此 Amazon 只需要存储您的公有密钥即可。公有密钥不具私密性，不能用于签署请求。多区域 API 请求必须使用非对称签名，例如 Amazon S3 多区域接入点。

以下步骤介绍了使用 SigV4a 计算签名的一般过程：

1. 待签字符串取决于请求类型。例如，使用 HTTP 授权标头或查询参数进行身份验证时，可以使用请求元素组合来创建待签字符串。对于 HTTP POST 请求，请求中的 POST 策略是您签名的字符串。
2. 签名密钥通过一系列计算从 Amazon 秘密访问密钥派生而来，每个步骤的结果都输入到下一个步骤中。最后一步将生成密钥对。
3. 当 Amazon 服务收到使用 SigV4a 签名的请求时，Amazon 仅使用密钥对的公共部分来验证签名。如果签名有效，则请求将经过身份验证，再由服务加以处理。系统将拒绝签名无效的请求。

有关用于多区域 API 请求的 SigV4a 的更多信息，请参阅 GitHub 上的 sigv4a-signing-examples 项目。

何时签署请求

编写自定义代码来将 API 请求发送给 Amazon 时，必须包含用于签署请求的代码。您可能需要编写自定义代码，原因如下：

- 您正在使用的编程语言没有对应的 Amazon 开发工具包。
- 您需要完全控制将请求发送给 Amazon 的方式。

API 请求使用 Amazon SigV4 验证访问权限，而 Amazon SDK 和 Amazon CLI 使用您提供的访问密钥对您的请求进行身份验证。有关使用 Amazon SDK 和 Amazon CLI 进行身份验证的更多信息，请参阅其他资源。

为什么签署请求

签名过程通过以下方式帮助保护请求：

- 验证请求者的身份

  经过身份验证的请求需要您使用访问密钥（访问密钥 ID、秘密访问密钥）创建的签名。如果您使用的是临时安全凭证，则签名计算还需要安全令牌。有关更多信息，请参阅 Amazon 安全凭证以编程方式访问。
- 保护传输中的数据

  为了防止传输时请求被篡改，一些请求元素将用于计算请求的哈希（摘要），得到的哈希值将包括在请求中。在 Amazon Web Services 服务 收到请求时，它将使用相同信息计算哈希，并将其与您的请求中包括的哈希值进行匹配。如果值不匹配，Amazon 将拒绝请求。
- 防止潜在的反演攻击

  在大多数情况下，请求必须在请求中的时间戳的 5 分钟内到达 Amazon。否则，Amazon 将拒绝该请求。

Amazon Sigv4 可以在 HTTP 授权标头中表示，也可以作为 URL 中的查询字符串表示。有关更多信息，请参阅 身份验证方法。

其他资源

- 有关不同服务的 Sigv4 签名过程的更多信息，请参阅 请求签名示例。

- 要为 Amazon CLI 的编程访问配置凭证，请参阅《Amazon Command Line Interface User Guide》中的 Authentication and access credentials。
- Amazon SDK 包括 GitHub 上用于签署 Amazon API 请求的源代码。有关代码示例，请参阅Amazon 示例存储库中的示例项目。
  - 适用于 .NET 的 Amazon SDK – AWS4Signer.cs
  - 适用于 C++ 的 Amazon SDK – AWSAuthV4Signer.cpp
  - 适用于 Go 的 Amazon SDK – sigv4.go
  - 适用于 Java 的 Amazon SDK – BaseAws4Signer.java
  - 适用于 JavaScript 的 Amazon SDK – signature-v4
  - 适用于 PHP 的 Amazon SDK – SignatureV4.php
  - Amazon SDK for Python (Boto) – signers.py
  - 适用于 Ruby 的 Amazon SDK – signer.rb

Amazon API 请求签名的元素

> **Important**
>
> 除非您使用 Amazon SDK 或 CLI，否则您必须编写代码来计算在请求中提供身份验证信息的签名。Amazon 签名版本 4 中的签名计算可能是一项复杂的任务，我们建议您尽可能使用 Amazon SDK 或 CLI。

每个使用 Signature Version 4 签名的 HTTP/HTTPS 请求都必须包含这些元素。

元素

- 终端节点规范
- Action
- 操作参数
- 日期
- 身份验证信息

终端节点规范

指定您要向其发送请求的端点的 DNS 名称。此名称通常包含服务代码和区域。例如，us-east-1 区域的 Amazon DynamoDB 端点为 dynamodb.us-east-1.amazonaws.com。

对于 HTTP/1.1 请求，您必须使用 Host 标头。对于 HTTP/2 请求，您可以使用 :authority 标头或 Host 标头。仅使用 :authority 标头以符合 HTTP/2 规范。并非所有服务都支持 HTTP/2 请求。

Action

为服务指定 API 操作。例如，DynamoDB CreateTable 操作或 Amazon EC2 DescribeInstances 操作。

有关每项服务支持的操作，请参阅 服务授权参考。

操作参数

指定请求中指定的操作的参数。每个 Amazon API 操作都有一组必备参数和可选参数。API 版本通常是必需参数。

有关 API 操作支持的参数，请参阅服务的 API 参考。

日期

指定请求的日期和时间。在请求中包括日期和时间有助于防止第三方拦截您的请求并稍后重新提交。您在凭证范围中指定的日期必须与您请求的日期匹配。

时间戳必须采用 UTC 表示，并具有以下 ISO 8601 格式：YYYYMMDDTHHMMSSZ。例如 20220830T123600Z。请勿在时间戳中包含毫秒。

您可以使用 date 标头或 x-amz-date 标头，或将 x-amz-date 作为查询参数包含在内。如果无法找到 x-amz-date 标头，则需要查找 date 标头。

身份验证信息

您发送的每个请求都必须包含以下信息。Amazon 使用这些信息来确保请求的有效性和真实性。

- 算法 – 您在签名过程中使用的算法。
  - SigV4 – 使用 AWS4-HMAC-SHA256 对 HMAC-SHA256 哈希算法指定 Signature Version 4。
  - SigV4a – 使用 AWS4-ECDSA-P256-SHA256 指定 ECDSA-P256-SHA-256 哈希算法。

- 凭证 – 通过将您的访问密钥 ID 和凭证范围组件串联起来而形成的字符串。
  - SigV4 – 凭证范围包括访问密钥 ID、YYYYMMDD 格式的日期、区域代码、服务代码和 aws4_request 终止字符串，用斜杠 (/) 分隔。区域代码、服务代码和终止字符串必须使用小写字符。

    ```
    AKIAIOSFODNN7EXAMPLE/YYYYMMDD/region/service/aws4_request
    ```

  - SigV4a – 凭证范围包括 YYYYMMDD 格式的日期、服务名称和 aws4_request 终止字符串，用斜杠 (/) 分隔。请注意，凭证范围不包括区域，因为该区域包含在单独的标头 X-Amz-Region-Set 中。

    ```
    AKIAIOSFODNN7EXAMPLE/YYYYMMDD/service/aws4_request
    ```

- 已签名标头 – 签名中要包含的 HTTP 标头，用分号 (;) 分隔。例如 host;x-amz-date。

  对于 SigV4a，您必须包含一个区域集标头，该标头指定请求将在哪一组区域中生效。标头 X-Amz-Region-Set 被指定为逗号分隔值的列表。下面的示例显示一个区域标头，该标头允许在 us-east-1 和 us-west-1 区域中进行请求。

  ```
  X-Amz-Region-Set=us-east-1,us-west-1
  ```

  您可以在区域中使用通配符 (*) 来指定多个区域。在下面的示例中，标头允许在 us-west-1 和 us-west-2 中进行请求。

  ```
  X-Amz-Region-Set=us-west-*
  ```

- 签名 – 代表计算得到的签名的十六进制编码字符串。您必须使用您在 Algorithm 参数中指定的算法来计算签名。

有关更多信息，请参阅 身份验证方法。

身份验证方法

> **Important**
>
> 除非您使用 Amazon SDK 或 CLI，否则您必须编写代码来计算在请求中提供身份验证信息的签名。Amazon 签名版本 4 中的签名计算可能是一项复杂的任务，我们建议您尽可能使用 Amazon SDK 或 CLI。

您可以使用以下方法之一快速传递身份验证信息：

HTTP 授权标头

HTTP Authorization 标头是验证请求的最常用方法。所有 REST API 操作（使用 POST 请求的基于浏览器的上传除外）都需要此标头。

以下示例显示了 SigV4 和 SigV4a 的 Authorization 标头值。为便于阅读，此示例中添加了换行符。在您的代码中，该标头必须是连续的字符串。算法和凭证之间没有逗号，但是，必须使用逗号分隔其他元素。

Example SigV4

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
SignedHeaders=host;range;x-amz-date,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

Example SigV4a

```
Authorization: AWS4-ECDSA-P256-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20130524/s3/aws4_request,
SignedHeaders=host;range;x-amz-date;x-amz-region-set,
Signature=fe5f80f77d5fa3beca038a248ff027d0445342fe2855ddc963176630326f1024
```

下表介绍了上述示例中授权标头值的各个组成部分：

组件	说明
授权	用于计算签名的算法。 • SigV4 – 使用 AWS4-HMAC-SHA256 。此字符串标识 Amazon SigV4（AWS4）和 HMAC-SHA256 算法。 • SigV4a – 使用 AWS4-ECDSA-P256-SHA256 。此字符串标识 Amazon SigV4（AWS4）和 ECDSA-P256-SHA-256 算法。
凭证	您的访问密钥 ID 和范围信息。

组件	说明

	• SigV4 – 包括用于计算签名的日期、区域和服务。该字符串具有以下形式：
	
	`<your-access-key-id>/<date>/ <aws-region>/<aws-service>/ aws4_request`
	• SigV4a – 包括用于计算签名的日期和服务。该字符串具有以下形式：
	
	`<your-access-key-id>/<date>/ <aws-service>/aws4_request`
	
	<date> 值使用 YYYYMMDD 格式指定。当发送请求到 Amazon S3 时，<aws-service> 值为 S3。

SignedHeaders	用于计算签名的请求标头的分号分隔列表。该列表仅包含标头名称，并且标头名称必须为小写。例如：`host;range;x-amz-date` 对于 SigV4a，您必须包含一个区域集标头，该标头指定请求将在哪一组区域中生效。标头 X-Amz-Region-Set 被指定为逗号分隔值的列表。
签名	256 位签名以 64 个小写十六进制字符表示。例如：`fe5f80f77d5fa3beca038a248ff 027d0445342fe2855ddc9631766 30326f1024` 请注意，签名计算因所选择的传输有效负载的选项而异。

查询字符串参数

您可以使用查询字符串在 URL 中完全表达请求。在这种情况下，您可以使用查询参数来提供请求信息，包括身份验证信息。由于请求签名是 URL 的一部分，因此这类 URL 通常称为预签名 URL。您

可以使用预签名 URL 在 HTML 中嵌入可单击链接，该链接的有效期最长 7 天。有关更多信息，请参阅《Amazon S3 API Reference》中的 Authenticating Requests: Using Query Parameters (Amazon Signature Version 4)。

以下示例显示了 SigV4 和 SigV4a 的预签名 URL。为便于阅读，此示例中添加了换行符：

Example SigV4

```
https://s3.amazonaws.com/amzn-s3-demo-bucket/test.txt ?
X-Amz-Algorithm=AWS4-HMAC-SHA256 &
X-Amz-Credential=<your-access-key-id>/20130721/<region>/s3/aws4_request &
X-Amz-Date=20130721T201207Z &
X-Amz-Expires=86400 &
X-Amz-SignedHeaders=host &X-Amz-Signature=<signature-value>
```

Example SigV4a

```
http://s3.amazonaws.com/amzn-s3-demo-bucket/test.txt ?
X-Amz-Algorithm=AWS4-ECDSA-P256-SHA256 &
X-Amz-Credential=<your-access-key-id>/20240721/s3/aws4_request &
X-amz-Region-Set=<regionset> &
X-Amz-Date=20240721T201207Z &
X-Amz-Expires=86400 &
X-Amz-SignedHeaders=host;x-amz-region-set &
X-Amz-Signature=<signature-value>
```

> **Note**
>
> URL 中的 X-Amz-Credential 值显示“/”字符只是为了方便阅读。实际上，它应该编码为 %2F。例如：
> &X-Amz-Credential=<your-access-key-id>%2F20130721%2Fus-east-1%2Fs3%2Faws4_request

下表介绍了 URL 中提供身份验证信息的查询参数。

查询字符串参数名称	说明
X-Amz-Algorithm	Amazon 签名的版本和用于计算签名的算法。

查询字符串参数名称	说明
	• SigV4 – 使用 AWS4-HMAC-SHA256 。此字符串标识 Amazon SigV4（AWS4）和 HMAC-SHA256 算法。 • SigV4a – 使用 AWS4-ECDSA-P256-SHA256 。此字符串标识 Amazon SigV4（AWS4）和 ECDSA-P256-SHA-256 算法。
X-Amz-Credential	除了访问密钥 ID 外，该参数还提供签名有效的范围。该值必须与您在签名计算中使用的范围匹配（后续部分将对此进行说明）。 • SigV4 – 该参数值的一般形式如下：<your-access-key-id>/<date>/<Amazon Region>/<Amazon-service>/aws4_request 例如：AKIAIOSFODNN7EXAMPLE/20130721/us-east-1/s3/aws4_request • SigV4a – 该参数值的一般形式如下：<your-access-key-id>/<date>/<Amazon-service>/aws4_request 例如：AKIAIOSFODNN7EXAMPLE/20130721/s3/aws4_request SigV4a 的区域在区域集标头 X-Amz-Region-Set 中定义。 有关 Amazon 区域字符串的列表，请参阅《Amazon General Reference》中的 Regional Endpoints。

查询字符串参数名称	说明
X-Amz-Region-Set	请求将在其中生效的区域集。标头 x-amz-region-set 被指定为逗号分隔值的列表。
X-Amz-Date	日期和时间格式必须遵循 ISO 8601 标准，并且必须按照 yyyyMMddTHHmmssZ 格式进行格式化。例如，如果日期和时间是“08/01/2016 15:32:41.982-700”，则必须先将其转换为 UTC（协调世界时），然后以“20160801T223241Z”形式提交。
X-Amz-Expires	提供生成的预签名 URL 的有效时间段（以秒为单位）。例如，86400（24 小时）。该值是一个整数。您可以设置的最小值为 1，最大值为 604800（七天）。预签名 URL 的有效期长达七天，因为您在签名计算中使用的签名密钥的有效期最长为七天。
X-Amz-SignedHeaders	列出用于计算签名的标头。签名计算中需要以下标头： • HTTP 主机标头。 • 您计划添加到请求的任何 x-amz-* 标头。 • 对于 SigV4a，需要使用 X-Amz-Region-Set 来指定可以发出请求的区域。 为了提高安全性，您应该签署计划在请求中包含的所有请求标头。
X-Amz-Signature	提供签名以验证您的请求。该签名必须与服务计算的签名相匹配；否则服务会拒绝该请求。例如，733255ef022bec3f2a8701cd61d4b371f3f28c9f193a1f02279211d48d5193d7 以下部分将介绍签名计算：

查询字符串参数名称	说明
X-Amz-Security-Token	如果使用来自 STS 服务的凭证，则为可选凭证参数。

创建已签名的 Amazon API 请求

> **Important**
>
> 如果您使用 Amazon SDK（请参阅示例代码和库）或 Amazon Command Line Interface（Amazon CLI）工具向 Amazon 发送 API 请求，则可以跳过本部分，因为 SDK 和 CLI 客户端会使用您提供的访问密钥来验证您的请求。除非您有充分的理由不这样做，否则我们建议您始终使用 SDK 或 CLI。
>
> 在支持多个签名版本的区域中，手动签名请求意味着您必须指定使用的签名版本。当您向多区域访问点提供请求时，SDK 和 CLI 会自动切换为使用签名版本 4A，而无需进行其他配置。

您可以使用 Amazon SigV4 签名协议为 Amazon API 请求创建签名的请求。

1. 根据请求详细信息创建规范请求。
2. 使用 Amazon 凭证计算签名。
3. 将此签名作为授权标头添加到请求。

然后 Amazon 复制此过程并验证签名，相应地授予或拒绝访问权限。

要了解如何使用 Amazon SigV4 对 API 请求进行签名，请参阅 请求签名示例。

下表介绍了创建签名请求过程中使用的函数。您需要为这些函数实现代码。如需了解更多信息，请参阅 Amazon 软件开发工具包中的代码示例。

函数	描述
`Lowercase()`	将字符串转换为小写。
`Hex()`	base-16 编码的小写形式。
`SHA256Hash()`	安全哈希算法（SHA）加密哈希函数。

函数	描述
`HMAC-SHA256()`	使用 SHA256 算法和提供的签名密钥计算 HMAC。这是您使用 SigV4 签名的最终签名。
`ECDSA-Sign`	使用基于公私密钥加密的非对称签名计算椭圆曲线数字签名算法（ECDSA）签名。
`KDF(K, Label, Context, L)`	处于计数器模式的 NIST SP800-108 KDF 使用 NIST SP 800-108r1 中定义的 PRF 函数 HMAC-SHA256。
`Oct2Int(byte[ ])`	ANSI X9.62 中描述的整数函数的八位字节。
`Trim()`	删除所有前导空格或尾随空格。

函数	描述
`UriEncode()`	URI 对每个字节进行编码。UriEncode() 必须强制执行以下规则： • URI 对除非预留字符之外的所有字节进行编码：“A”-“Z”、“a”-“z”、“0”-“9”、“-”、“.”、“_”和“~”。 • 空格字符是预留字符，必须编码为“%20”（而不是“+”）。 • 每个 URI 编码字节由“%”和该字节的两位十六进制值组成。 • 十六进制值中的字母必须为大写，例如“%1A”。 • 对除对象键名称之外的所有位置的正斜杠字符“/”进行编码。例如，如果对象键名称是 `photos/Jan/sample.jpg` ，则不会对键名称中的正斜杠进行编码。 **Important** 由于底层 RFC 中的实现差异和相关歧义，您开发平台提供的标准 UrienCode 函数可能无法正常工作。建议您编写自己的自定义 UrienCode 函数，以确保编码能够正常工作。 要查看 Java 中的 UriEncode 函数示例，请参阅 GitHub 网站上的 Java Utilities。

> **Note**
>
> 签署请求时，您可以使用 Amazon SigV4 或 Amazon SigV4a。两者之间的关键区别取决于签名的计算方式。使用 SigV4a 时，区域集包含在待签字符串中，但不是凭证派生步骤的一部分。

使用临时安全凭证签名请求

您可使用 Amazon Security Token Service（Amazon STS）提供的临时安全凭证来签署请求，而不是使用长期凭证。

使用临时安全凭证时，必须将 `X-Amz-Security-Token` 添加至授权标头或将其包含在查询字符串中以保存会话令牌。某些服务会要求您将 `X-Amz-Security-Token` 添加至规范请求。其他服务仅会要求您在计算出签名后在末尾添加 `X-Amz-Security-Token`。有关具体要求，请查看每个 Amazon Web Services 服务的文档。

签名步骤摘要

创建规范请求

将请求的内容（主机、操作、标头等）组织为标准规范格式。规范请求是用于创建待签字符串的输入之一。有关创建规范请求的详细信息，请参阅Amazon API 请求签名的元素。

创建规范请求的哈希值

使用创建负载的哈希时所使用的相同算法来哈希规范请求。经过哈希处理的规范请求必须以小写十六进制字符串形式表示。

创建待签字符串

使用规范请求和额外信息（例如算法、请求日期、凭证范围和规范请求的哈希）创建待签字符串。

派生签名密钥

使用秘密访问密钥派生用于对请求进行签名的密钥。

计算签名

使用派生的签名密钥作为哈希密钥，对待签字符串执行加密哈希操作。

将签名添加至请求

将计算的签名添加到请求的 HTTP 标头或查询字符串中。

创建规范请求

要创建规范请求，请串联由换行符分隔的以下字符串。这有助于确保您计算出的签名能够与 Amazon 计算出的签名相匹配。

```
<HTTPMethod>\n
<CanonicalURI>\n
<CanonicalQueryString>\n
<CanonicalHeaders>\n
<SignedHeaders>\n
<HashedPayload>
```

- *HTTPMethod* – HTTP 方法，例如 GET、PUT、HEAD 和 DELETE。
- *CanonicalUri*：绝对路径组件 URI 的 URI 编码版本，以域名后面的 / 开头，直至字符串结尾处，或者如果包含查询字符串参数，则直至问号字符（？）。如果绝对路径为空，则使用正斜杠字符（/）。以下示例中的 URI /amzn-s3-demo-bucket/myphoto.jpg 是绝对路径，并且您无需在绝对路径中对 / 进行编码：

  ```
  http://s3.amazonaws.com/amzn-s3-demo-bucket/myphoto.jpg
  ```

- *CanonicalQueryString* – URI 编码的查询字符串参数。您可以单独对每个名称和值进行 URI 编码。您还必须按键名称的字母顺序对规范查询字符串中的参数进行排序。编码后进行排序。以下 URI 示例中的查询字符串是：

  ```
  http://s3.amazonaws.com/amzn-s3-demo-bucket?prefix=somePrefix&marker=someMarker&max-keys=2
  ```

  规范查询字符串如下所示（为便于阅读，此示例中添加了换行符：）：

  ```
  UriEncode("marker")+"="+UriEncode("someMarker")+"&"+
  UriEncode("max-keys")+"="+UriEncode("20") + "&" +
  UriEncode("prefix")+"="+UriEncode("somePrefix")
  ```

  当请求针对子资源时，相应的查询参数值将为空字符串（""）。例如，以下 URI 标识了 amzn-s3-demo-bucket 存储桶上的 ACL 子资源：

```
http://s3.amazonaws.com/amzn-s3-demo-bucket?acl
```

在这种情况下，CanonicalQueryString 将为：

```
UriEncode("acl") + "=" + ""
```

如果 URI 不包含 ?，则请求中没有查询字符串，并且您需要将规范查询字符串设置为空字符串（""）。您仍然需要包含换行符 ("\n")。

- *CanonicalHeaders*：请求标头及其值的列表。各个标头名称和值对用换行符（"\n"）分隔。以下是 CanonicalHeader 的示例：

```
Lowercase(<HeaderName1>)+":"+Trim(<value>)+"\n"
Lowercase(<HeaderName2>)+":"+Trim(<value>)+"\n"
...
Lowercase(<HeaderNameN>)+":"+Trim(<value>)+"\n"
```

CanonicalHeaders 列表必须包含以下内容：

- HTTP host 标头。
- 如果请求中存在 Content-Type 标头，则必须将其添加到 *CanonicalHeaders* 列表中。
- 此外，还必须添加计划在请求中包含的所有 x-amz-* 标头。例如，如果您使用临时安全凭证，则请求中必须包含 x-amz-security-token。您必须将此标头添加到 *CanonicalHeaders* 列表中。
- 对于 SigV4a，您必须包含一个区域集标头，该标头指定请求将在哪一组区域中生效。标头 X-Amz-Region-Set 被指定为逗号分隔值的列表。下面的示例显示一个区域标头，该标头允许在 us-east-1 和 us-west-1 区域中进行请求。

  ```
  X-Amz-Region-Set=us-east-1,us-west-1
  ```

  您可以在区域中使用通配符（*）来指定多个区域。在下面的示例中，标头允许在 us-west-1 和 us-west-2 中进行请求。

  ```
  X-Amz-Region-Set=us-west-*
  ```

> **Note**
>
> x-amz-content-sha256 标题是 Amazon S3 Amazon 请求所必需的。它将提供请求负载的哈希。如果不包含有效负载，则必须提供空字符串的哈希值。

每个标头名称必须：

- 使用小写字符。
- 按字母顺序显示。
- 后跟冒号（：）。

对于值，您必须：

- 去除任何前导空格或尾随空格。
- 将连续空格转换为单个空格。
- 使用逗号分隔多值标头的值。
- 签名中必须包含 host 标头（HTTP/1.1）或 :authority 标头（HTTP/2）以及任何 x-amz-* 标头。签名中也可以包含其他标准标头，例如 content-type。

上一部分介绍了本示例中使用的 Lowercase() 和 Trim() 函数。

以下是示例 CanonicalHeaders 字符串。标头名称为小写且已排序。

```
host:s3.amazonaws.com
x-amz-content-sha256:e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855
x-amz-date:20130708T220855Z
```

> **Note**
>
> 为了计算授权签名，只有主机以及任何 x-amz-* 标头为必填项；但是，为了防止数据篡改，您应该考虑在签名计算中包含其他标头。
> 请勿包含会在复杂系统传输中频繁更改的逐跳标头。这包括所有由代理、负载均衡器和分布式系统中的节点改变的易失性传输标头，包括 connection、x-amzn-trace-id、user-agent、keep-alive、transfer-encoding、TE、trailer、upgrade、proxy-authorization 和 proxy-authenticate。

- *SignedHeaders*：按字母顺序排序、以分号分隔的小写请求标头名称列表。列表中的请求标头与您在 CanonicalHeaders 字符串中包含的标头相同。对于前面的示例，*SignedHeaders* 的值如下：

```
host;x-amz-content-sha256;x-amz-date
```

- *HashedPayload* – 使用 HTTP 请求正文中的负载作为哈希函数的输入创建的字符串。此字符串使用小写十六进制字符。

```
Hex(SHA256Hash(<payload>))
```

如果请求中不包含有效负载，则计算空字符串的哈希值，例如，当使用 GET 请求检索对象时，有效负载中没有任何内容。

```
Hex(SHA256Hash(""))
```

> **Note**
>
> 对于 Amazon S3，请在构造规范请求时包含文字字符串 UNSIGNED-PAYLOAD，并在发送请求时设置与 x-amz-content-sha256 标头值相同的值。
> Hex(SHA256Hash("UNSIGNED-PAYLOAD"))

创建规范请求的哈希值

使用创建负载的哈希时所使用的相同算法来创建规范请求的哈希（摘要）。经过哈希处理的规范请求必须以小写十六进制字符串形式表示。

创建待签字符串

要创建待签字符串，请串联以下由换行符分隔的以下字符串。请勿使用换行符作为此字符串的结尾。

```
Algorithm \n
RequestDateTime \n
CredentialScope  \n
HashedCanonicalRequest
```

- *Algorithm* – 用于创建规范请求的哈希的算法。

  - SigV4 – 使用 AWS4-HMAC-SHA256 指定 HMAC-SHA256 哈希算法。
  - SigV4a – 使用 AWS4-ECDSA-P256-SHA256 指定 ECDSA-P256-SHA-256 哈希算法。
- *RequestDateTime* – 在凭证范围内使用的日期和时间。该值是采用 ISO 8601 格式的当前 UTC 时间（例如 20130524T000000Z）。
- *CredentialScope*：凭证范围，将生成的签名限制在指定的区域和服务范围内。
  - SigV4 – 凭证包括访问密钥 ID、YYYYMMDD 格式的日期、区域代码、服务代码和 aws4_request 终止字符串，用斜杠（/）分隔。区域代码、服务代码和终止字符串必须使用小写字符。字符串具有以下格式：YYYYMMDD/region/service/aws4_request。
  - SigV4a – 凭证包括 YYYYMMDD 格式的日期、服务名称和 aws4_request 终止字符串，用斜杠（/）分隔。请注意，凭证范围不包括区域，因为该区域包含在单独的标头 X-Amz-Region-Set 中。字符串具有以下格式：YYYYMMDD/service/aws4_request。
- *HashedCanonicalRequest*：上一步中计算出的规范请求的哈希。

以下是要签名的字符串的示例。

```
"<Algorithm>" + "\n" +
timeStampISO8601Format + "\n" +
<Scope> + "\n" +
Hex(<Algorithm>(<CanonicalRequest>))
```

派生签名密钥

要派生签名密钥，请选择以下过程之一来计算 SigV4 或 SigV4a 的签名密钥。

派生 SigV4 的签名密钥

要派生 SigV4 的签名密钥，请使用 Amazon 秘密访问密钥作为初始哈希操作的密钥，对请求日期、区域和服务执行一系列加密哈希操作（HMAC）。

对于每个步骤，使用所需的密匙和数据调用哈希函数。每次调用哈希函数的结果都会变成下一次调用哈希函数的输入。

以下示例说明了如何派生本过程下一部分中使用的 SigningKey，并说明了输入的串联和哈希顺序。HMAC-SHA256 是用于对数据进行哈希处理的哈希功能，如下所示。

```
DateKey = HMAC-SHA256("AWS4"+"<SecretAccessKey>", "<YYYYMMDD>")
DateRegionKey = HMAC-SHA256(<DateKey>, "<aws-region>")
```

```
DateRegionServiceKey = HMAC-SHA256(<DateRegionKey>, "<aws-service>")
SigningKey = HMAC-SHA256(<DateRegionServiceKey>, "aws4_request")
```

必填项

- Key – 包含您的秘密访问密钥的字符串。
- Date – 包含在凭证范围中使用的日期的字符串，格式为 YYYYMMDD。
- Region – 包含区域代码的字符串（例如，us-east-1）。

  有关区域字符串的列表，请参阅 Amazon Web Services 一般参考 中的 Regional Endpoints。
- Service – 包含服务代码的字符串（例如，ec2）。
- 在上一步中创建的要签名的字符串。

要派生 SigV4 的签名密钥

1. 串联 "AWS4" 和秘密访问密钥。使用密钥和数据调用哈希函数，并将连接的字符串作为密钥，而日期字符串作为数据。

   ```
   DateKey = hash("AWS4" + Key, Date)
   ```
2. 使用密钥和数据调用哈希函数，并将上一次调用的结果作为密钥，而区域字符串作为数据。

   ```
   DateRegionKey = hash(kDate, Region)
   ```
3. 使用密钥和数据调用哈希函数，并将上一次调用的结果作为密钥，而服务字符串作为数据。

   服务代码由服务定义。您可以在 Amazon Pricing CLI 中使用 get-products 返回服务的服务代码。

   ```
   DateRegionServiceKey = hash(kRegion, Service)
   ```
4. 使用密钥和数据调用哈希函数，并将上一次调用的结果作为密钥，而“aws4_request”作为数据。

   ```
   SigningKey = hash(kService, "aws4_request")
   ```

派生 SigV4a 的签名密钥

要为 SigV4a 创建签名密钥，请使用以下过程从秘密访问密钥派生密钥对。有关此派生实现的示例，请参阅 Amazon 客户端身份验证的 C99 库实现

```
n = [NIST P-256 elliptic curve group order]
G = [NIST P-256 elliptic curve base point]
label = "AWS4-ECDSA-P256-SHA256"

akid = [Amazon access key ID as a UTF8 string]
sk = [Amazon secret access Key as a UTF8 Base64 string]

input_key = "AWS4A" || sk
count = 1
while (counter != 255) {
  context = akid || counter // note: counter is one byte
  key = KDF(input_key, label, context, 256)
  c = Oct2Int(key)
  if (c > n - 2) {
    counter++
  } else {
    k = c + 1   // private key
    Q = k * G   // public key
  }
}

if (c < 255) {
  return [k, Q]
} else {
  return FAILURE
}
```

计算签名

派生签名密钥后，计算要添加到请求中的签名。此过程因您使用的签名版本而不同。

要计算 SigV4 的签名

1. 将上一次调用的结果作为密钥，要签名的字符串作为数据来调用哈希函数。使用派生的签名密钥作为此操作的哈希密钥。结果是作为二进制值的签名。

   ```
   signature = hash(SigningKey, string-to-sign)
   ```

2. 将签名从二进制转换为十六进制表示形式，使用小写字符。

要计算 SigV4a 的签名

1. 使用数字签名算法（ECDSA P-256），对您在上一步中创建的要签名的字符串进行签名。用于此签名的密钥是从上述秘密访问密钥派生的非对称私有密钥。

```
signature = base16(ECDSA-Sign(k, string-to-sign))
```

2. 将签名从二进制转换为十六进制表示形式，使用小写字符。

将签名添加至请求

将计算出的签名添加到您的请求中。

Example示例：授权标头

SigV4

以下示例显示了使用 Amazon SigV4 的 `DescribeInstances` 操作的 `Authorization` 标头。为便于阅读，此示例已使用换行符编排过格式。在您的代码中，这必须是连续的字符串。算法和 `Credential` 之间没有逗号。但是，必须使用逗号分隔其他元素。

```
Authorization: AWS4-HMAC-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request,
SignedHeaders=host;x-amz-date,
Signature=calculated-signature
```

SigV4a

以下示例显示了使用 Amazon SigV4a 的 `CreateBucket` 操作的授权标头。为便于阅读，此示例已使用换行符编排过格式。在您的代码中，这必须是连续的字符串。算法和凭证之间没有逗号。但是，必须使用逗号分隔其他元素。

```
Authorization: AWS4-ECDSA-P256-SHA256
Credential=AKIAIOSFODNN7EXAMPLE/20220830/s3/aws4_request,
SignedHeaders=host;x-amz-date;x-amz-region-set,
Signature=calculated-signature
```

Example示例：请求在查询字符串中使用身份验证参数

SigV4

以下示例显示了对包含身份验证信息的使用 Amazon SigV4 的 `DescribeInstances` 操作的查询。为便于阅读，此示例已使用换行符编排过格式，而非 URL 编码。在您的代码中，查询字符串必须是采用 URL 编码的连续字符串。

```
https://ec2.amazonaws.com/?
Action=DescribeInstances&
Version=2016-11-15&
X-Amz-Algorithm=AWS4-HMAC-SHA256&
X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20220830/us-east-1/ec2/aws4_request&
X-Amz-Date=20220830T123600Z&
X-Amz-SignedHeaders=host;x-amz-date&
X-Amz-Signature=calculated-signature
```

SigV4a

以下示例显示了对包含身份验证信息的使用 Amazon SigV4a 的 `CreateBucket` 操作的查询。为便于阅读，此示例已使用换行符编排过格式，而非 URL 编码。在您的代码中，查询字符串必须是采用 URL 编码的连续字符串。

```
https://ec2.amazonaws.com/?
Action=CreateBucket&
Version=2016-11-15&
X-Amz-Algorithm=AWS4-ECDSA-P256-SHA256&
X-Amz-Credential=AKIAIOSFODNN7EXAMPLE/20220830/s3/aws4_request&
X-Amz-Region-Set=us-west-1&
X-Amz-Date=20220830T123600Z&
X-Amz-SignedHeaders=host;x-amz-date;x-amz-region-set&
X-Amz-Signature=calculated-signature
```

请求签名示例

Amazon 签名请求的以下示例向您介绍如何使用 SigV4 对在没有 Amazon SDK 或 Amazon 命令行工具的情况下发送的请求进行签名。

使用 HTTP POST 进行基于浏览器的 Amazon S3 上传

对请求进行身份验证：基于浏览器的上传介绍 Amazon S3 在收到请求时用来计算签名的签名和相关信息。

示例：使用 HTTP POST 进行基于浏览器的上传（使用 Amazon 签名版本 4）提供更多信息，其中包含示例 POST 策略和可用于上传文件的表单。示例策略和虚拟凭证向您介绍工作流程以及生成的签名和策略哈希。

VPC Lattice 经过身份验证的请求

签名版本 4（SigV4）经过身份验证的请求示例提供 Python 和 Java 示例，显示了如何在使用和不使用自定义拦截器的情况下执行请求签名。

对 Amazon Translate 使用签名版本 4

元宇宙中的实时翻译展示了如何构建能够生成近乎实时的翻译解决方案的应用程序。这种语音到语音翻译器解决方案在事件流编码中使用 Amazon SigV4 来生成实时转录。

对 Neptune 使用签名版本 4

示例：搭配使用 Python 和签名版本 4 签名连接到 Neptune 介绍如何使用 Python 向 Neptune 发出签名请求。此示例包括使用访问密钥或临时凭证的变体。

向 Amazon Glacier 签署 HTTP 请求

流式处理 API 的签名计算示例介绍为上传档案（发布档案）创建签名的详细信息，上传档案（发布档案）是 Amazon Glacier 中的两个流式处理 API 之一。

向 Amazon SWF 发出 HTTP 请求

向 Amazon SWF 发出 HTTP 请求显示了向 Amazon SWF 发出的 JSON 请求的标题内容。

Amazon OpenSearch Service 中流式处理 API 的签名计算

使用适用于 PHP 的 Amazon SDK 版本 3 对 Amazon OpenSearch Service 搜索请求签名包括如何向 Amazon OpenSearch Service 发送已签名的 HTTP 请求的示例。

Amazon 示例存储库中的示例项目

以下示例项目显示了如何签署请求，以便使用 Python、Node.js、Java、C#、Go 和 Rust 等常见语言向 Amazon 服务发出 Rest API 请求。

签名版本 4a 项目

sigv4-signing-examples 项目提供了如何使用 SigV4 签署请求，以便使用 Python、Node.js、Java、C#、Go 和 Rust 等常用语言向 Amazon Web Services 服务 发出 Rest API 请求的示例。

sigv4a-signing-examples 项目提供了签署多区域 API 请求的示例，例如 Amazon S3 中的多区域访问点。

发布到 Amazon IoT Core

使用 HTTPS 协议发布到 Amazon IoT Core 的 Python 代码提供了如何使用 HTTPS 协议和 Amazon SigV4 身份验证向 Amazon IoT Core 发布信息的指导。它有两个参考实施，一个在 Python 中，另一个在 NodeJS 中。

使用 HTTPS 协议发布到 Amazon IoT Core 的 .Net Framework 应用程序提供了如何使用 HTTPS 协议和 Amazon SigV4 身份验证向 Amazon IoT Core 发布信息的指导。该项目还包括一个 .NET Core 等效实施。

排查 Amazon API 请求的签名版本 4 签名问题

> **Important**
>
> 除非您使用 Amazon SDK 或 CLI，否则您必须编写代码来计算在请求中提供身份验证信息的签名。SigV4 签名计算可能十分复杂，我们建议您尽可能使用 Amazon SDK 或 CLI。

在开发可创建已签名请求的代码时，您可能会从 Amazon Web Services 服务收到 HTTP 403 `SignatureDoesNotMatch` 错误。此类错误表示您对 Amazon 发出的 HTTP 请求中的签名值与 Amazon Web Services 服务 计算出的签名不一致。当权限不允许调用者发出请求时，系统会返回 HTTP 401 `Unauthorized` 错误。

出现以下情况时，API 请求可能会返回错误：

- API 请求未签名，并且 API 请求使用的是 IAM 身份验证。
- 用于签署请求的 IAM 证书不正确或无权调用该 API。
- 已签名 API 请求的签名与 Amazon 服务计算出的签名不一致。
- API 请求标头不正确。

> **Note**
>
> 请首先将签名协议从 Amazon 签名版本 2（SigV2）更新为 Amazon 签名版本 4（SigV4），然后再探索其他错误解决方案。Amazon S3 等服务和区域不再支持 Sigv2 签名。

可能的原因

- 凭证错误
- 规范请求和签名字符串错误
- 凭证范围错误
- 密钥签名错误

凭证错误

确保 API 请求是使用 Sigv4 签署的。如果 API 请求未签名，则可能会收到以下错误消息：`Missing Authentication Token`。添加缺失的签名 并重新发送请求。

确认访问密钥和私有密钥的身份验证凭证准确无误。如果访问密钥不正确，则可能会收到以下错误消息：`Unauthorized`。确保用于签署请求的实体有权提出请求。有关更多信息，请参阅 排查访问被拒绝错误消息。

规范请求和签名字符串错误

如果您在 创建规范请求的哈希值 或 创建待签字符串 中计算的规范请求有误，则服务执行的签名验证步骤将会失败，并显示以下错误消息：

```
The request signature we calculated does not match the signature you provided
```

Amazon 服务收到已签名的请求后，将会重新计算签名。如果两者的值存在差异，则签名不一致。将已签名请求的规范请求和字符串与错误消息中的值进行比较。如果两者有任何差异，请修改签名过程。

> **Note**
>
> 您还可以验证您没有通过修改标头或请求的代理发送请求。

Example规范请求示例

```
GET                                                    -------- HTTP method
/                                                      -------- Path. For API stage
 endpoint, it should be /{stage-name}/{resource-path}
                                                       -------- Query string key-
value pair. Leave it blank if the request doesn't have a query string.
content-type:application/json                          -------- Header key-value
 pair. One header per line.
```

```
host:0123456789.execute-api.us-east-1.amazonaws.com      -------- Host and x-amz-date
 are required headers for all signed requests.
x-amz-date:20220806T024003Z

content-type;host;x-amz-date                              -------- A list of signed
 headers
d167e99c53f15b0c105101d468ae35a3dc9187839ca081095e340f3649a04501        -------- Hash
 of the payload
```

要验证密钥是否与访问密钥 ID 匹配，您可以使用已知的有效实施对其进行测试。例如，使用 Amazon SDK 或 Amazon CLI 向 Amazon 发出请求。

API 请求标头

当授权标头为空，凭证密钥或签名缺失或不正确，标头不是以算法名称开头，或者键值对不包含等号时，您会收到以下错误之一：

- 授权标头不能为空。
- 授权标头需要“Credential”参数。
- 授权标头需要“Signature”参数。
- 签名在授权标头中包含无效的 key=value 对（缺少等号）。

请确保您在 计算签名 中添加的 SigV4 授权标头包含正确的凭证密钥，同时包含使用 HTTP 日期或 `x-amz-date` 标头的请求日期。

如果您收到 IncompleteSignatureException 错误且签名的构造正确，则可以通过计算客户端请求中授权标头的 SHA-256 哈希值和 B64 编码来验证授权标头在传输到 Amazon Web Services 服务 的过程中未被修改。

1. 获取您在请求中发送的授权标头。您的授权标头与以下示例类似：

   ```
   Authorization: AWS4-HMAC-SHA256
   Credential=AKIAIOSFODNN7EXAMPLE/20130524/us-east-1/s3/aws4_request,
   SignedHeaders=host;range;x-amz-date,
   Signature=example-generated-signature
   ```

2. 计算授权标头的 SHA-256 哈希值。

   ```
   hashSHA256(rawAuthorizationHeader) = hashedAuthorizationHeader
   ```

3. 将经过哈希处理的授权标头编码为 Base64 格式。

```
base64(hashedAuthorizationHeader) = encodedHashedAuthorizationHeader
```

4. 将刚才计算的经过哈希处理和编码的字符串与您在错误消息中收到的字符串进行比较。您的错误消息应类似于以下示例：

```
com.amazon.coral.service#IncompleteSignatureException:
The signature contains an in-valid key=value pair (missing equal-sign)
in Authorization header (hashed with SHA-256 and encoded with Base64):
'9c574f83b4b950926da4a99c2b43418b3db8d97d571b5e18dd0e4f3c3ed1ed2c'.
```

- 如果两个哈希值不同，则授权标头的某些部分在传输过程中更改。此更改可能是由于您的网络或客户端处理程序附加签名的标头或以某种方式更改授权标头。
- 如果两个哈希值匹配，则您在请求中发送的授权标头与 Amazon 收到的内容相匹配。查看您收到的错误消息，确定问题是否是凭证或签名不正确的结果。本页面的其他部分将介绍这些错误。

凭证范围错误

您在 创建待签字符串 中创建的凭证范围可将签名限定为特定的日期、区域和服务。此字符串具有以下格式：

```
YYYYMMDD/region/service/aws4_request
```

> **Note**
>
> 如果您使用的是 Sigv4a，则该区域未包括在凭证范围内。

日期

如果凭证范围未指定与 x-amz-date 标头相同的日期，则签名验证步骤将失败，并显示以下错误消息：

```
Date in Credential scope does not match YYYYMMDD from ISO-8601 version of date from
 HTTP
```

如果请求指定了未来时间，则签名验证步骤将失败，并显示以下错误消息：

```
Signature not yet current: date is still later than date
```

如果请求已过期，则签名验证步骤将失败，并显示以下错误消息：

```
Signature expired: date is now earlier than date
```

区域

如果凭证范围未指定与请求相同的区域，则签名验证步骤将失败，并显示以下错误消息：

```
Credential should be scoped to a valid Region, not region-code
```

服务

如果凭证范围未指定与 host 标头相同的服务，则签名验证步骤将失败，并显示以下错误消息：

```
Credential should be scoped to correct service: 'service'
```

终止字符串

如果凭证范围没有以 aws4_request 结尾，则签名验证步骤将失败，并显示以下错误消息：

```
Credential should be scoped with a valid terminator: 'aws4_request'
```

密钥签名错误

由于不正确地派生签名密钥或使用密码术而导致的错误更难排查。验证规范字符串和待签字符串正确无误后，还可以检查是否存在以下某个问题：

- 秘密访问密钥与您指定的访问密钥 ID 不匹配。
- 您的密钥派生代码存在问题。

要验证密钥是否与访问密钥 ID 匹配，您可以使用已知的有效实施对其进行测试。例如，使用 Amazon SDK 或 Amazon CLI 向 Amazon 发出请求。有关示例，请参阅请求签名示例。

IAM JSON 策略参考

本节介绍了 IAM 中的 JSON 策略的元素、变量和评估逻辑的详细语法、描述和示例。有关更多一般信息，请参阅 JSON 策略概述。

此参考包括以下部分。

- IAM JSON 策略元素参考 - 详细了解您在创建策略时可以使用的元素。查看其他策略示例，并了解条件、支持的数据类型以及在各种服务中的使用方式。
- 策略评估逻辑 - 本部分介绍 Amazon 请求、如何对其进行身份验证，以及 Amazon 如何使用策略来确定对资源的访问权限。
- IAM JSON 策略语言的语法 - 本部分介绍用于在 IAM 中创建策略的语言的正式语法。
- 工作职能的 Amazon 托管策略 – 本部分列出直接映射到 IT 行业常见工作职能的所有 Amazon 托管策略。使用这些策略可以授予执行特定工作职能中的预期任务所需要的权限。这些策略将许多服务的权限整合到一个策略中。
- Amazon 全局条件上下文密钥 - 本部分包含可用于限制 IAM policy 中的权限的所有 Amazon 全局条件键的列表。
- IAM 和 Amazon STS 条件上下文密钥 – 本部分包含可用于限制 IAM policy 中的权限的所有 IAM 和 Amazon STS 全局条件键的列表。
- Amazon 服务的操作、资源和条件键 — 本部分提供了您可以在 IAM policy 中用作权限的所有 Amazon API 操作。它还包括可用于进一步优化请求的特定于服务的条件键。

IAM JSON 策略元素参考

JSON 策略文档由元素组成。这些元素按照在策略中使用的大致顺度列出。元素的顺序并不重要，例如，`Resource` 元素可以在 `Action` 元素之前出现。您无需在策略中指定任何 `Condition` 元素。要详细了解 JSON 策略文档的一般结构和目的，请参阅 JSON 策略概述。

一些 JSON 策略元素相互排斥。这意味着，您不能创建同时使用两个元素的策略。例如，您不能在同一策略语句中同时使用 `Action` 和 `NotAction`。相互排斥的其他对包括 `Principal`/`NotPrincipal` 和 `Resource`/`NotResource`。

策略中所包含的具体内容因产品而异，其取决于产品所提供的操作、包含的资源类型等。当您针对特定产品编写策略时，参考该产品的示例将为您提供帮助。如需获取支持 IAM 的所有服务的列表，以及阐述这些服务的 IAM 和策略的文档链接，请参阅 使用 IAM 的 Amazon 服务。

当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

主题

- IAM JSON 策略元素：Version
- IAM JSON 策略元素：Id
- IAM JSON 策略元素：Statement
- IAM JSON 策略元素：Sid
- IAM JSON 策略元素：Effect
- Amazon JSON 策略元素：Principal
- Amazon JSON 策略元素：NotPrincipal
- IAM JSON 策略元素：Action
- IAM JSON 策略元素：NotAction
- IAM JSON 策略元素：Resource
- IAM JSON 策略元素：NotResource
- IAM JSON 策略元素：Condition
- IAM policy 元素：变量和标签
- IAM JSON 策略元素：支持的数据类型

IAM JSON 策略元素：Version

> **消除歧义的说明**
>
> 此 `Version` JSON 策略元素与策略版本 不同。`Version` 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您对 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。如果您在寻找与托管策略适用的多版本支持相关的信息，请参阅the section called “IAM 策略版本控制”。

`Version` 策略元素指定用于处理策略的语言语法规则。要使用所有可用策略功能，请将以下 `Version` 元素包含在所有策略中的 `Statement` 元素之外。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

IAM 支持以下 Version 元素值：

- 2012-10-17. 这是策略语言的当前版本，您应始终包含一个 Version 元素并将其设置为 2012-10-17。否则，您将无法使用此版本中引入的策略变量 等功能。
- 2008-10-17. 这是策略语言的早期版本。您可能会在较旧的现有策略中看到此版本。请勿将此版本用于任何新策略或正在更新的任何现有策略。较新的功能（如策略变量）将不适用于您的策略。例如，系统不会将 ${aws:username} 等变量识别为变量，而是将其视为策略中的文本字符串。

IAM JSON 策略元素：Id

Id 元素指定策略的可选标识符。不同产品使用的 ID 也不尽相同。ID 可在基于资源的策略中使用，但不能在基于身份的策略中使用。

对于允许您设置 ID 元素的产品，我们建议您使用 UUID (GUID) 值，或将 UUID 与 ID 合并，以确保唯一性。

JSON

```
{
  "Version":"2012-10-17",
  "Id": "cd3ad3d9-2776-4ef1-a904-4c229d1642ee",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

> **Note**
>
> 某些 Amazon 产品（例如，Amazon SQS 或 Amazon SNS）可能需要该元素并要求其具有唯一性。有关编写策略的特定于服务的信息，请参阅您使用的服务的文档。

IAM JSON 策略元素：Statement

`Statement` 元素为策略的主要元素。该元素为必填项。`Statement` 元素可以包含一条语句或由单独语句组成的数组。每条单独的语句块必须用大括号 { } 括起来。对于多条语句，数组必须用方括号 [ ] 括起来。

```
"Statement": [{...},{...},{...}]
```

在下列示例所列举的策略中，在单一 `Statement` 元素内包含一个有三份声明的数组。(该策略允许您在 Amazon S3 控制台中访问您自己的“主文件夹”。) 该策略包含 `aws:username` 变量，在策略评估期间它将被替换为请求中的用户名称。有关更多信息，请参阅 简介。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {"StringLike": {"s3:prefix": [
        "",
        "home/",
        "home/${aws:username}/"
      ]}}
```

```
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}",
        "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}/*"
      ]
    }
  ]
}
```

IAM JSON 策略元素：Sid

您可以提供 Sid（语句 ID）作为策略语句的可选标识符。您可以为语句数组中的每个语句指定 Sid 值。您可以使用 Sid 值作为策略声明的描述。在允许您指定 ID 元素的产品中，例如 SQS 和 SNS，Sid 值正是策略文件 ID 的子 ID。在 IAM 中，Sid 值必须在 JSON 策略中是唯一的。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExampleStatementID",
      "Effect": "Allow",
      "Action": "s3:ListAllMyBuckets",
      "Resource": "*"
    }
  ]
}
```

Sid 元素支持 ASCII 大写字母 (A-Z)、小写字母 (a-z) 和数字 (0-9)。

IAM 不会公开在 IAM API 中的 Sid。您无法基于此 ID 检索特定声明。

> **Note**
>
> 某些 Amazon 产品（例如，Amazon SQS 或 Amazon SNS）可能需要该元素并要求其具有唯一性。有关编写策略的特定于服务的信息，请参阅您使用的服务的文档。

IAM JSON 策略元素：Effect

`Effect` 元素是必需具备的元素，用于指定声明所产生的结果是“允许”还是“显式拒绝”。`Effect` 的有效值为 `Allow` 和 `Deny`。`Effect` 值区分大小写。

```
"Effect":"Allow"
```

在默认情况下，将拒绝访问资源。如要允许访问资源，您必须将 `Effect` 元素设置为 `Allow`。如要置换“允许”结果 (例如，置换否则会生效的“允许”结果)，您需要将 `Effect` 元素设置为 `Deny`。有关更多信息，请参阅 策略评估逻辑。

Amazon JSON 策略元素：Principal

在基于资源的 JSON 策略中使用 `Principal` 元素指定允许或拒绝访问资源的主体。

您必须使用基于资源的策略中的 `Principal` 元素。包括 IAM 在内的多项服务支持基于资源的策略。IAM 中唯一基于资源的策略类型是角色信任策略。在 IAM 角色中，在角色的信任策略中使用 `Principal` 元素来指定可担任该角色的对象。对于跨账户访问，您必须指定受信任账户的 12 位标识符。要了解您信任区域之外的账户（受信任的企业或账户）中的主体是否有权承担您的角色，请参阅什么是 IAM Access Analyzer？。

> **Note**
>
> 创建角色后，您可以将账户更改为 "*"，允许所有人担任该角色。如果执行此操作，我们强烈建议您通过其他方法 (如将访问只限定为特定 IP 地址的 `Condition` 元素) 限制能够访问该角色的用户。不要将角色的访问权限开放给所有人！

支持基于资源的策略的其他资源示例包括 Amazon S3 存储桶或 Amazon KMS key。

无法在基于身份的策略中使用 `Principal` 元素。基于身份的策略是附加到 IAM 身份（用户、群体或角色）的权限策略。在这些策略中，附加了策略的身份即是主体的身份。

主题

- 如何指定主体
- Amazon Web Services 账户 主体
- IAM 角色主体
- 角色会话主体
- OIDC 联合身份验证主体
- SAML 联合身份验证主体
- IAM 用户主体
- IAM Identity Center 主体
- Amazon STS 联合用户主体
- Amazon 服务主体
- 选择加入区域的 Amazon 服务主体
- 所有主体
- 更多信息

如何指定主体

您可以在基于资源的策略的 `Principal` 元素中或在支持主体的条件密钥中指定主体。

您可以在策略中指定以下任意主体：

- Amazon Web Services 账户 和根用户
- IAM 角色
- 角色会话
- IAM 用户
- 联合用户主体
- Amazon 服务
- 所有主体

您无法在策略（例如基于资源的策略）中将用户组标识为主体，因为组与权限相关，与身份验证无关，并且主体是经过身份验证的 IAM 实体。

您可以使用数组在以下部分中为每个主体类型指定多个主体。数组可以采用一个或多个值。如果您在该元素中指定多个主体，则表示您向每个主体授予权限。这是逻辑 OR 而不是逻辑 AND，因为您一次被认证为一个主体。如果包含多个值，请使用方括号（[和]）并用逗号分隔数组的每个条目。以下示例策略定义了 123456789012 账户或 555555555555 账户的权限。

```
"Principal" : {
"AWS": [
  "123456789012",
  "555555555555"
  ]
}
```

> **Note**
>
> 您不能使用通配符匹配一部分主体名称或 ARN。

Amazon Web Services 账户 主体

您可以在基于资源策略的 `Principal` 元素中或支持主体的条件键中指定 Amazon Web Services 账户 标识符。这将权限委派给账户。当您允许访问其他账户时，该账户中的管理员必须为该账户中身份（IAM 用户或角色）授予访问权限。在指定 Amazon Web Services 账户 时，您可以使用账户 ARN (arn:aws:iam::*account-ID*:root) 或 `"AWS":` 前缀后加账户 ID 构成的简略格式。

例如，给定账户 ID 为 `123456789012` 的情况下，您可以使用以下任一方法来在 `Principal` 元素中指定账户：

```
"Principal": { "AWS": "arn:aws:iam::123456789012:root" }
```

```
"Principal": { "AWS": "123456789012" }
```

账户 ARN 和缩短的账户 ID 的行为方式相同。两者都向账户委派权限。在 `Principal` 元素中使用账户 ARN 并不会将权限限制为该账户的根用户。

> **Note**
>
> 当您保存包含缩短账户 ID 的基于资源的策略时，服务可能会将其转换为主体 ARN。这不会更改策略的功能。

某些 Amazon 服务支持其他用于指定账户主体的选项。例如，Amazon S3 允许您使用以下格式指定规范用户 ID：

```
"Principal": { "CanonicalUser":
 "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be" }
```

您还可以指定多个 Amazon Web Services 账户（或规范用户 ID）作为使用数组的主体。例如，您可以使用所有三种方法在存储桶策略中指定主体。

```
"Principal": {
  "AWS": [
    "arn:aws:iam::123456789012:root",
    "999999999999"
  ],
  "CanonicalUser": "79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be"
}
```

IAM 角色主体

您可以在支持主体的基于资源的策略或条件密钥的 `Principal` 元素中指定 IAM 角色主体 ARN。IAM 角色是身份。在 IAM 中，身份是您可以向其分配权限的资源。角色信任另一个身份验证来担任该角色。这包含 Amazon 中的主体或来自外部身份提供程序 (IdP) 的用户。当主体或身份代入角色时，他们会收到具有代入角色权限的临时安全凭证。当他们使用这些凭证在 Amazon 中执行操作时，将变为角色会话主体。

当您在基于资源的策略中指定角色主体时，主体的有效权限受限制该角色权限的任何策略类型的限制。这包括会话策略和权限边界。有关如何评估角色会话的有效权限的更多信息，请参阅 策略评估逻辑。

要指定 `Principal` 元素中的角色 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:iam::Amazon-account-ID:role/role-name" }
```

> **Important**
>
> 如果角色信任策略的 `Principal` 元素中包含指向特定 IAM 角色的 ARN，在保存策略时该 ARN 将转换为该角色的唯一主体 ID。如果有人希望通过删除并重新创建角色来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为 IAM 在显示信任策略时使用反向转换回角色 ARN。但是，如果您删除角色，这种关系即被打破。即使您重新创建角色，策略也不再适用。因为新角色拥有新的主体 ID，与信任策略中存储的 ID 不匹配。发生这种情况时，主体 ID 会出现在基于资源的策略中，因为 Amazon 无法将其映射回有效的 ARN。最终

结果是，如果您删除并重新创建了信任策略的 Principal 元素所引用的角色，您必须编辑策略中的角色，用正确的 ARN 替换主体 ID。当您保存策略时，ARN 会再次转换为该角色新的主体 ID。有关更多信息，请参阅了解 Amazon 的对已删除 IAM 角色的处理策略。

IAM 联合身份用户 — IAM 用户使用 aws:PrincipalArn 操作联合身份，为 IAM 用户生成联合身份用户会话主体。当您使用此密钥时，角色会话主体将根据所担任的角色的 ARN 而不是所生成的会话的 ARN 授予权限。由于 Amazon 不将条件密钥 ARN 转换为 ID，如果您删除该角色，然后创建一个具有相同名称的新角色，则授予该角色 ARN 的权限将继续存在。基于身份的策略类型，例如权限边界或会话策略，不限制在 Principal 元素中使用带通配符 (*) 的 aws:PrincipalArn 条件键授予权限，除非基于身份的策略包含显式拒绝。

角色会话主体

您可以在支持主体的基于资源的策略或条件密钥的 Principal 元素中指定角色会话。当主体或身份代入角色时，他们会收到具有代入角色权限的临时安全凭证。当他们使用这些凭证，在 Amazon 中执行操作时，将变为角色会话主体。

您用于角色会话主体的格式取决于用来代入角色的 Amazon STS 操作。

> **Important**
>
> Amazon 建议尽可能在策略中使用 IAM 角色主体而不是角色会话主体。必要时使用 Condition 语句和条件键进一步缩小访问范围。

要在 Principal 元素中指定角色会话主体 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:assumed-role/role-name/role-session-name" }
```

此外，管理员可以设计一个流程来控制角色会话的发放方式。例如，他们可以为用户提供一键式解决方案，以创建可预测的会话名称。如果管理员执行此操作，则您可以在策略或条件密钥中使用角色会话主体。否则，您可以在 aws:PrincipalArn 条件键中将角色 ARN 指定为主体。如何将角色指定为主体可以更改所生成的会话的有效权限。有关更多信息，请参阅 IAM 角色主体。

OIDC 联合身份验证主体

OIDC 联合身份验证主体是指使用符合 OIDC 标准的 IDP (也称为 OpenID Provider (OP)) 提供的 JSON 网络令牌 (JWT) 调用 Amazon STS AssumeRoleWithWebIdentity API 以请求临时 Amazon

凭证时使用的主体。OIDC 联合身份验证主体可以代表 Amazon 账户中的 OIDC IDP，也可以代表 4 个内置身份提供商：Login with Amazon、Google、Facebook 和 Amazon Cognito。

已从 OIDC IDP 签发 JWT 的用户、工作负载或系统可使用 JWT 来调用 `AssumeRoleWithWebIdentity`，以请求 IAM 角色的临时 Amazon 安全凭证，该 IAM 角色已配置为信任“签发 JWT 的 OIDC IDP”。只要符合 Amazon STS 列出的要求，JWT 可以是 ID 令牌、访问令牌，也可以是通过任何其他方法交付的 JWT 令牌。有关更多信息，请参阅常见场景以及通过 OIDC 提供商请求凭证。

在您的角色信任策略中使用此主体类型，以允许或拒绝使用 Amazon Web Services 账户 中存在的 OIDC IDP 或四个内置 IDP 中的一个来调用 `AssumeRoleWIthWebIdentity` 的权限。要在角色信任策略的 `Principal` 元素中指定 OIDC 联合身份验证主体 ARN，请使用以下四种内置 OIDC IDP 的格式之一：

```
"Principal": { "Federated": "cognito-identity.amazonaws.com" }
```

```
"Principal": { "Federated": "www.amazon.com" }
```

```
"Principal": { "Federated": "graph.facebook.com" }
```

```
"Principal": { "Federated": "accounts.google.com" }
```

在使用您添加到账户的 OIDC 提供商（如 GitHub）时，您需要在角色的信任策略中指定提供商的 ARN。通过此配置，您可以编写 IAM 策略，专门控制通过自定义身份提供商验证的用户的访问。

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:oidc-provider/full-OIDC-identity-provider-URL" }
```

例如，如果 GitHub 是受信任的网络身份提供商，则角色信任策略 `Principal` 元素中的 OIDC 角色会话 ARN 使用以下格式：

```
"Principal": { "Federated": "arn:aws:iam::AWS-account-ID:oidc-provider/
tokens.actions.githubusercontent.com" }
```

有关更多信息，请参阅在 Amazon Web Services 中配置 OpenID Connect。

角色信任策略以外的策略类型不支持 OIDC 联合身份验证主体。

SAML 联合身份验证主体

SAML 联合身份验证主体是指在调用 Amazon STS `AssumeRoleWithSAML` API 时使用 SAML 断言来请求临时 Amazon 凭证的主体。您可以使用您的 SAML 身份提供程序 (IdP) 登录，然后使用此操作来代入 IAM 角色。与 `AssumeRoleWithWebIdentity` 类似，`AssumeRoleWithSAML` 也不需要 Amazon 凭证进行身份验证。相反，用户首先通过其 SAML 身份提供程序进行身份验证，然后使用其 SAML 断言进行 `AssumeRoleWithSAML` API 调用，或被重定向至 Amazon 登录/SAML 页面以登录 Amazon Web Services 管理控制台。有关哪些主体可以使用此操作代入角色的更多信息，请参阅 比较 Amazon STS 凭证。

在您的角色信任策略中使用此主体类型，可基于受信任的 SAML 身份提供程序来允许或拒绝权限。要指定角色信任策略 `Principal` 元素中 SAML 身份角色会话的 ARN，请采用以下格式：

```
"Principal": { "Federated": "arn:aws:iam::Amazon-account-ID:saml-provider/provider-name" }
```

IAM 用户主体

您可以在支持主体的基于资源的策略或条件键的 `Principal` 元素中指定 IAM 用户。

> **Note**
>
> 在 `Principal` 元素中，Amazon Resource Name (ARN) 的用户名部分区分大小写。

```
"Principal": { "AWS": "arn:aws:iam::Amazon-account-ID:user/user-name" }
```

```
"Principal": {
  "AWS": [
    "arn:aws:iam::Amazon-account-ID:user/user-name-1",
    "arn:aws:iam::Amazon-account-ID:user/user-name-2"
  ]
}
```

当在 `Principal` 元素中指定用户时，不能使用通配符 (*) 表示“所有用户”。主体必须始终指明特定用户。

> **Important**
>
> 如果角色信任策略的 `Principal` 元素中包含指向特定 IAM 用户的 ARN，则在保存策略时该 IAM 将 ARN 转换为该用户的唯一主体 ID。如果有人希望通过删除并重新创建用户来提升特权，这样有助于减轻此类风险。您通常不会在控制台中看到这个 ID，因为显示信任策略时它还会反向转换为用户的 ARN。但是，如果您删除角色，这种关系即被打破。即使您重新创建用户，策略也不再适用。这是因为：新用户拥有新的主体 ID，该 ID 与信任策略中存储的 ID 不匹配。发生这种情况时，主体 ID 会出现在基于资源的策略中，因为 Amazon 无法将其映射回有效的 ARN。结果是，如果您删除并重新创建了信任策略的 `Principal` 元素所引用的用户，您必须编辑角色，用正确的 ARN 替换目前不正确的主体 ID。当您保存策略时，IAM 会再次将 ARN 转换为该用户新的主体 ID。

IAM Identity Center 主体

在 IAM Identity Center 中，必须将基于资源的策略中的主体定义为 Amazon Web Services 账户 主体。要指定访问权限，请引用条件块中设置的权限的角色 ARN。有关详细信息，请参阅《IAM Identity Center 用户指南》中的引用资源策略、Amazon EKS 和 Amazon KMS 中的权限集。

Amazon STS 联合用户主体

您可以在支持主体的基于资源的策略或条件密钥的 `Principal` 元素中指定 federated user sessions（联合身份用户会话）。

> **Important**
>
> Amazon 建议您限制 Amazon STS 联合用户会话的使用。而应使用 IAM 角色。

Amazon STS 联合用户主体是通过调用长期 IAM 凭证的 `GetFederationToken` 操作创建的。联合用户权限是调用 `GetFederationToken` 的主体与作为参数传递给 `GetFederationToken` API 的会话策略的交集。

在 Amazon 中，IAM 用户或 Amazon Web Services 账户根用户 可以使用长期访问密钥进行身份验证。有关哪些主体可以使用此操作联合身份的更多信息，请参阅 比较 Amazon STS 凭证。

- IAM 联合用户 – IAM 用户使用 `GetFederationToken` 操作联合身份，为该 IAM 用户生成联合用户会话。

- 联合根用户 – 根用户使用 GetFederationToken 操作联合身份，为该根用户生成联合用户会话。

当 IAM 用户或根用户通过使用此操作的 Amazon STS 请求临时凭证时，他们开启临时的联合身份会话。本会话的 ARN 基于联合身份的原始身份。

要指定 Principal 元素中联合身份用户会话的 ARN，请采用以下格式：

```
"Principal": { "AWS": "arn:aws:sts::AWS-account-ID:federated-user/user-name" }
```

Amazon 服务主体

您可以在支持主体的基于资源的策略或条件密钥的 Principal 元素中指定 Amazon 服务。一个服务主体是服务的标识符。

可以由 Amazon 服务担任的 IAM 角色称为服务角色。服务角色必须包括信任策略。信任策略是附加到角色的基于资源的策略，这些策略定义了可担任该角色的主体。某些服务角色具有预定义的信任策略。但有些情况下，您必须在信任策略中指定服务主体。IAM policy 中的服务主体不能是 "Service": "*"。

> **Important**
>
> 服务主体的标识符包括服务名称，通常采用以下格式：
> *service-name*.amazonaws.com
> 但是，某些服务可能使用以下格式代替常用格式或作为常用格式的补充：
> *service-name*.amazonaws.com

服务主体由服务定义。可以通过以下方式找到某些服务的服务主体：打开 使用 IAM 的 Amazon 服务，检查服务在 Service-linked role（服务相关角色）列中是否具有 Yes，然后打开 Yes（是）链接以查看该服务的服务相关角色文档。查找该服务的服务相关角色权限部分，查看服务主体。

以下示例显示了一个可以附加到服务角色的策略。该策略可启用两个服务（Amazon ECS 和 Elastic Load Balancing）以担任该角色。然后，这些服务可执行由分配给该角色 (未显示) 的权限策略授权执行的任何任务。要指定多个服务主体，不用指定两个 Service 元素；您可以只使用一个该元素。实际上，您可以将一组多个服务主体作为单个 Service 元素的值。

```
"Principal": {
    "Service": [
        "ecs.amazonaws.com",
```

```
        "elasticloadbalancing.amazonaws.com"
    ]
}
```

选择加入区域的 Amazon 服务主体

您可以在多个 Amazon 区域以及您必须选择加入的其中一些区域启动资源。有关您必须选择加入的区域的完整列表，请参阅 Amazon Web Services 一般参考 指南中的管理 Amazon 区域。

当选择加入区域中的 Amazon 服务在同一区域内提出请求时，服务主体名称格式被标识为其服务主体名称的非区域化版本：

service-name.amazonaws.com

当选择加入区域中的 Amazon 服务向另一个区域内提出跨区域请求时，服务主体名称格式被标识为其服务主体名称的区域化版本：

service-name.*{region}*.amazonaws.com

例如，您有一个 Amazon SNS 主题位于区域 ap-southeast-1，一个 Amazon S3 桶位于选择加入区域 ap-east-1。您想要配置 S3 桶通知以向 SNS 主题发布消息。要允许 S3 服务向 SNS 主题发布消息，您必须通过该主题的基于资源的访问策略授予 S3 服务主体 sns:Publish 权限。

如果您在主题访问策略中指定了 S3 服务主体的非区域化版本 s3.amazonaws.com，则从桶向主题发出的 sns:Publish 请求将失败。以下示例在 SNS 主题访问策略的 Principal 策略元素中指定了非区域化 S3 服务主体。

```
"Principal": { "Service": "s3.amazonaws.com" }
```

由于桶位于选择加入区域，并且请求是在同一区域之外发出的，因此 S3 服务主体显示为区域化服务主体名称 s3.ap-east-1.amazonaws.com。当选择加入区域中的 Amazon 服务向另一个区域发出请求时，您必须使用区域化服务主体名称。指定区域化服务主体名称后，如果桶向位于另一个区域的 SNS 主题发出 sns:Publish 请求，则请求将成功。以下示例在 SNS 主题访问策略的 Principal 策略元素中指定了区域化 S3 服务主体。

```
"Principal": { "Service": "s3.ap-east-1.amazonaws.com" }
```

只有指定了区域化服务主体名称，资源策略或基于服务主体的允许列表才能成功处理从一个选择加入区域到另一个区域的跨区域请求。

> **Note**
>
> 对于 IAM 角色信任策略，我们建议使用非区域化服务主体名称。IAM 资源是全局性的，因此可以在任何区域使用相同的角色。

所有主体

您可以在基于资源的策略的 `Principal` 元素或在支持主体的条件键中使用通配符（`*`）指定所有主体。使用 基于资源的策略 授予权限和条件键来限制策略语句的条件。

> **Important**
>
> 我们强烈建议不要在带有 `Allow` 效果的基于资源的策略的 `Principal` 元素中使用通配符（`*`），除非您打算授予公共或匿名访问权限。否则，请在 `Principal` 元素中指定预期的主体、服务或 Amazon 账户，然后进一步将访问权限限制在 `Condition` 元素内。对于 IAM 角色信任策略尤其如此，因为它们允许其他主体成为您账户中的主体。

对于基于资源的策略，使用带 `Allow` 效果的通配符（`*`）向所有用户（包括匿名用户）授予访问权限（公共访问权限）。对于账户中的 IAM 用户和角色主体，不需要其他权限。对于其他账户中的主体，他们还必须在其账户中拥有基于身份的权限，以允许他们访问您的资源。这称为 cross-account access（跨账户访问）。

对于匿名用户，以下元素是等效的：

```
"Principal": "*"
```

```
"Principal" : { "AWS" : "*" }
```

您不能使用通配符匹配一部分主体名称或 ARN。

以下示例说明了基于资源的策略（而非 Amazon JSON 策略元素：NotPrincipal）可用于显式拒绝除 `Condition` 元素中指定的主体之外的所有主体。应将此策略添加至 Amazon S3 存储桶。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "UsePrincipalArnInsteadOfNotPrincipalWithDeny",
      "Effect": "Deny",
      "Action": "s3:*",
      "Principal": "*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/*",
        "arn:aws:s3:::amzn-s3-demo-bucket"
      ],
      "Condition": {
        "ArnNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::444455556666:user/user-name"
        }
      }
    }
  ]
}
```

更多信息

有关更多信息，请参阅下列内容：

- 《Amazon Simple Storage Service 用户指南》中的 存储桶策略示例
- 《Amazon Simple Notification Service 开发人员指南》中的 Amazon SNS 示例策略
- 《Amazon Simple Queue Service 开发商指南》中的 Amazon SQS 策略示例
- 《Amazon Key Management Service 开发人员指南》中的 密钥策略
- 《Amazon Web Services 一般参考》中的 账户标识符
- OIDC 联合身份验证

Amazon JSON 策略元素：NotPrincipal

`NotPrincipal` 元素使用 `"Effect":"Deny"` 拒绝所有主体的访问权限（`NotPrincipal` 元素中指定的主体除外）。主体可以是 IAM 用户、Amazon STS 联合用户主体、IAM 角色、所担任的角色会话、Amazon Web Services 账户、Amazon 服务或其他主体类型。有关主体的更多信息，请参阅 Amazon JSON 策略元素：Principal。

`NotPrincipal` 必须与 `"Effect":"Deny"` 一起使用。不支持将其与 `"Effect":"Allow"` 一起使用。

> **Important**
>
> 我们不建议在基于资源的新策略中使用 NotPrincipal 作为安全和授权策略的一部分。如果您使用 NotPrincipal，则可能难以排查多策略类型的影响。我们建议改用带 ARN 条件运算符的 aws:PrincipalArn 上下文键。

关键点

- 对于部分 Amazon 服务（包括 VPC 端点），基于资源的策略也支持 NotPrincipal 元素。基于资源的策略是直接嵌入资源中的策略。您不能在基于 IAM 身份的策略或 IAM 角色信任策略中使用 NotPrincipal 元素。
- 如果基于资源的策略语句包含对附加了权限边界策略的 IAM 用户或角色具有 Deny 效果的 NotPrincipal 策略元素，则不要使用这种策略语句。具有 Deny 效果的 NotPrincipal 元素将始终拒绝任何附加了权限边界策略的 IAM 主体，无论 NotPrincipal 元素中指定的值为何。这会导致本来可以访问该资源的某些 IAM 用户或角色失去访问权限。我们建议更改基于资源的策略语句，以将 ArnNotEquals 条件运算符和 aws:PrincipalArn 上下文键结合使用来限制访问权限，而不是使用 NotPrincipal 元素。有关权限边界的信息，请参阅 IAM 实体的权限边界。
- 当您使用 NotPrincipal 时，还必须指定未被拒绝主体的账户 ARN。否则，策略可能会拒绝对包含主体的整个账户的访问。根据您的策略中包括的服务，Amazon 可能会先验证账户，然后验证用户。如果评估一个担任角色的用户（使用角色的某个人），Amazon 会相继验证账户、角色，最后才是担任角色的用户。担任角色的用户是由他们担任角色时指定的角色会话名称标识的。因此，我们强烈建议您明确包括用户账户的 ARN，或者包括角色的 ARN 以及包含角色的账户的 ARN。
- 服务控制策略（SCP）和资源控制策略（RCP）不支持 NotPrincipal 元素。

NotPrincipal 元素的替代方案

在 Amazon 中管理访问控制时，可能会遇到需要显式拒绝所有主体对某个资源的访问权限的情况，但您指定的一个或多个主体除外。Amazon 建议使用带有全局条件上下文键的 Deny 语句，以实现更精确的控制和更轻松的故障排除。以下示例显示了使用条件运算符（如 StringNotEquals 或 ArnNotEquals）拒绝所有主体的访问权限（条件元素中指定的主体除外）的其他方法。

使用 IAM 角色的示例场景

您可以使用带有 Deny 语句的基于资源的策略来防止所有 IAM 角色（条件元素中指定的角色除外）访问或操作您的资源。这种方法遵循 Amazon 安全原则，即显式拒绝始终优先于任何 allow 语句，并且有助于在整个 Amazon 基础设施中维护最低权限原则。

我们建议不要使用 NotPrincipal，而是使用带有全局条件上下文键和条件运算符（如 ArnNotEquals）的 Deny 语句，来显式允许 IAM 角色访问您的资源。以下示例使用 aws:PrincipalArn 显式允许 read-only-role 角色访问 Bucket_Account_Audit 文件夹中的 Amazon S3 存储桶。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyCrossAuditAccess",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::Bucket_Account_Audit",
        "arn:aws:s3:::Bucket_Account_Audit/*"
      ],
      "Condition": {
        "ArnNotEquals": {
          "aws:PrincipalArn": "arn:aws:iam::444455556666:role/read-only-role"
        }
      }
    }
  ]
}
```

使用服务主体的示例场景

您可以使用 Deny 语句来防止所有服务主体（Condition 元素中指定的主体除外）访问或操作您的资源。当您需要在 Amazon 环境中实施精细访问控制或在不同服务和应用程序之间建立安全边界时，这种方法特别有用。

我们建议不要使用 NotPrincipal，而是使用带有全局条件上下文键和条件运算符（如 StringNotEquals）的 Deny 语句，来显式允许服务主体访问您的资源。以下示例使用 aws:PrincipalServiceName 显式允许 Amazon CodeBuild 服务主体访问 BUCKETNAME 文件夹中的 Amazon S3 存储桶。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyNotCodeBuildAccess",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::BUCKETNAME",
        "arn:aws:s3:::BUCKETNAME/*"
      ],
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:PrincipalServiceName": "codebuild.amazonaws.com"
        }
      }
    }
  ]
}
```

IAM JSON 策略元素：Action

`Action` 元素描述将允许或拒绝的特定操作。声明必须包含 `Action` 或 `NotAction` 元素。每款 Amazon 服务各自拥有一套描述任务的操作，您可使用相应服务来执行所描述的任务。例如，Amazon S3 的操作列表可以在 Amazon Simple Storage Service 用户指南中的在策略中指定权限找到，Amazon EC2 操作列表可以在Amazon EC2 API 参考中找到，而适用于 Amazon Identity and Access Management 的操作列表可以在 IAM API 参考中找到。要查找其他服务的操作列表，请参阅 API 参考文档了解相关服务。

此外，Amazon 以 JSON 格式提供服务参考信息，以简化策略管理工作流程的自动化。借助服务参考信息，您可以从机器可读文件中跨 Amazon Web Services 服务访问可用操作、资源和条件键。有关更多信息，请参阅《服务授权参考》中的用于编程访问的简化 Amazon Web Services 服务信息。

通过将服务命名空间用作操作前缀（`iam`、`ec2`、`sqs`、`sns`、`s3` 等）并后跟允许或拒绝的操作名称来指定值。该名称必须与产品支持的操作相匹配。前缀和操作名称不区分大小写。例如，`iam:ListAccessKeys` 与 `IAM:listaccesskeys` 相同。以下示例显示用于不同服务的 `Action` 元素。

Amazon SQS 操作

```
"Action": "sqs:SendMessage"
```

Amazon EC2 操作

```
"Action": "ec2:StartInstances"
```

IAM 操作

```
"Action": "iam:ChangePassword"
```

Amazon S3 操作

```
"Action": "s3:GetObject"
```

您可以为 Action 元素指定多个值。

```
"Action": [ "sqs:SendMessage", "sqs:ReceiveMessage", "ec2:StartInstances",
 "iam:ChangePassword", "s3:GetObject" ]
```

可以使用多字符匹配通配符 (*) 和单字符匹配通配符 (?)，来授予对特定 Amazon 产品所提供所有操作的访问权限。例如，以下 Action 元素适用于所有 S3 操作。

```
"Action": "s3:*"
```

还可以将通配符（* 或 ?）作为操作名称的一部分使用。例如，下列 Action 元素适用于所有包含字符串 AccessKey 的 IAM 操作，包括 CreateAccessKey、DeleteAccessKey、ListAccessKeys 及 UpdateAccessKey。

```
"Action": "iam:*AccessKey*"
```

某些产品允许您限制可用的操作。例如，Amazon SQS 允许您只提供所有可能的 Amazon SQS 操作的一部分。在这种情况下，* 通配符不允许完全控制队列；而是仅允许控制您已共享的操作子集。有关更多信息，请参阅 Amazon Simple Storage Service 开发人员指南中的了解权限。

IAM JSON 策略元素：NotAction

NotAction 是与指定的操作列表之外 的所有内容显式匹配的高级策略元素。使用 NotAction 时只列出不应匹配的一些操作，而不是包括将匹配的长操作列表，因此生成的策略较短。NotAction 中指定的操作不受策略语句中 Allow 或 Deny 效果影响。这也就意味着，如果使用 Allow 效果，则允许未列出的所有适用操作或服务。此外，如果使用 Deny 效果，则拒绝此类未列出的操作或服务。将 NotAction 与 Resource 元素结合使用时，您需要提供策略的范围。这是 Amazon 确定哪些操作或服务适用的方式。有关更多信息，请参阅以下策略示例。

NotAction 与 Allow

可在包含 "Effect": "Allow" 的语句中使用 NotAction 元素来提供对 Amazon 服务中所有操作（在 NotAction 中指定的操作除外）的访问权限。您可以将它与 Resource 元素结合使用以提供策略的范围，从而将允许的操作限制为那些可在指定资源上执行的操作。

以下示例允许用户访问可在任何 S3 资源上执行的所有 Amazon S3 操作，但删除存储桶的操作除外。此策略也不允许其他服务中的操作，因为其他服务操作不适用于 S3 资源。

```
"Effect": "Allow",
"NotAction": "s3:DeleteBucket",
"Resource": "arn:aws:s3:::*",
```

有时您可能需要允许访问大量操作。使用 NotAction 元素可有效地修改语句，生成更短的操作列表。例如，由于 Amazon 提供了如此多的服务，您可能需要创建一个策略，以允许用户执行除访问 IAM 操作之外的所有操作。

下面的示例允许用户访问除 IAM 之外的 Amazon 服务中的所有操作。

```
"Effect": "Allow",
"NotAction": "iam:*",
"Resource": "*"
```

在同一语句或一个策略内的不同语句中使用 NotAction 元素和 "Effect": "Allow" 时要谨慎。NotAction 匹配未显式列出或适用指定资源的所有服务和操作，并且可能导致向用户授予超出您意图的更多权限。

NotAction 与 Deny

可在一个语句中将 NotAction 元素与 "Effect": "Deny" 结合使用，以拒绝访问列出的所有资源，在 NotAction 元素中指定的操作除外。此组合不允许列出的项目，而是显式拒绝未列出的操作。您仍必须允许您要允许的操作。

下面的条件示例在用户未使用 MFA 登录时拒绝其访问非 IAM 操作。如果用户已使用 MFA 登录，则 "Condition" 测试失败，最后的 "Deny" 语句将无效。但请注意，这不会向用户授予对任何操作的访问权限，只会明确拒绝除 IAM 操作之外的所有其他操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [{
        "Sid": "DenyAllUsersNotUsingMFA",
        "Effect": "Deny",
        "NotAction": "iam:*",
        "Resource": "*",
        "Condition": {"BoolIfExists": {"aws:MultiFactorAuthPresent": "false"}}
    }]
}
```

有关拒绝访问特定区域以外的操作（来自特定服务的操作除外）的示例策略，请参阅Amazon：根据请求的区域拒绝访问 Amazon。

IAM JSON 策略元素：Resource

IAM 策略语句中的 Resource 元素指定了该语句适用的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。

使用 Amazon 资源名称（ARN）指定资源。ARN 的格式取决于 Amazon Web Services 服务 和所引用的特定资源。尽管 ARN 格式各不相同，但您始终使用 ARN 来标识资源。有关 ARN 格式的更多信息，请参见 IAM ARN。有关如何指定资源的信息，请参阅您编写的资源声明所对应的产品文档。

> **Note**
>
> 有些 Amazon Web Services 服务 不允许您为单个资源指定操作。在这些情况下，您在 Action 或 NotAction 元素中列出的任何操作都适用于该服务中的所有资源。如果是这种情况，则可以在 Resource 元素中使用通配符（*）。

下列示例适用于特定的 Amazon SQS 队列。

```
"Resource": "arn:aws:sqs:us-east-2:account-ID-without-hyphens:queue1"
```

以下示例引用了 Amazon Web Services 账户 中名为 Bob 的 IAM 用户。

> **Note**
>
> 在 Resource 元素中，IAM 用户名区分大小写。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/Bob"
```

在资源 ARN 中使用通配符

您可在 ARN（用冒号分隔的部分）的各分段中使用通配符（* 和？）来表示：

- 字符的任意组合（*）
- 任何单个字符（？）

您可以在每个分段中使用多个 * 或？字符。如果 * 通配符是资源 ARN 分段的最后一个字符，则它可以扩展以匹配冒号边界以外的内容。我们建议您在 ARN 分段内使用通配符（* 和？），用冒号隔开。

> **Note**
>
> 不得在服务分段中使用可识别 Amazon 产品的通配符。有关 ARN 分段的更多信息，请参阅 使用 Amazon 资源名称（ARN）标识 Amazon 资源

以下示例引用了其路径为 /accounting 的所有 IAM 用户。

```
"Resource": "arn:aws:iam::account-ID-without-hyphens:user/accounting/*"
```

下列示例适用于特定 Amazon S3 存储桶内的所有项目。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*"
```

星号（*）字符可以展开以替换分段中的所有内容，包括像正斜杠（/）这样的字符，否则这些字符可能在给定服务命名空间中显示为分隔符。例如，请考虑以下 Amazon S3 ARN，因为相同的通配符扩展逻辑适用于所有服务。

```
"Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*/test/*"
```

ARN 中的通配符适用于存储桶中的以下所有对象，而不仅仅是列出的第一个对象。

```
amzn-s3-demo-bucket/1/test/object.jpg
amzn-s3-demo-bucket/1/2/test/object.jpg
amzn-s3-demo-bucket/1/2/test/3/object.jpg
amzn-s3-demo-bucket/1/2/3/test/4/object.jpg
amzn-s3-demo-bucket/1///test///object.jpg
amzn-s3-demo-bucket/1/test/.jpg
amzn-s3-demo-bucket//test/object.jpg
amzn-s3-demo-bucket/1/test/
```

考虑前一个列表中的最后两个对象。Amazon S3 对象名称可以常规分隔符正斜杠（/）字符开头或结尾。虽然 / 可作为分隔符，但在资源 ARN 中使用此字符时没有特定意义。它将被视为与任何其他有效字符相同。ARN 将不与以下对象匹配：

```
amzn-s3-demo-bucket/1-test/object.jpg
amzn-s3-demo-bucket/test/object.jpg
amzn-s3-demo-bucket/1/2/test.jpg
```

指定多个资源

您可以使用 ARN 数组在 Resource 元素中指定多个资源。下列示例适用于两个 DynamoDB 表。

```
"Resource": [
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/books_table",
    "arn:aws:dynamodb:us-east-2:account-ID-without-hyphens:table/magazines_table"
]
```

在资源 ARN 中使用策略变量

在 Resource 元素中，您可以在标识特定资源的 ARN 部分 (即，ARN 尾部) 中使用 JSON 策略变量。例如，您可以使用键 {aws:username} 作为资源 ARN 的一部分，以表示应包含当前用户的名称作为资源名称的一部分。下列示例显示如何在 {aws:username} 元素中使用 Resource 键。该策略允许访问匹配当前用户名称的 Amazon DynamoDB 表。

JSON

```
{
    "Version":"2012-10-17",
```

```
    "Statement": {
        "Effect": "Allow",
        "Action": "dynamodb:*",
        "Resource": "arn:aws:dynamodb:us-east-2:111122223333:table/
${aws:username}"
    }
}
```

有关 JSON 策略变量的更多信息，请参阅IAM policy 元素：变量和标签。

IAM JSON 策略元素：NotResource

NotResource 是高级策略元素，可明确匹配除指定资源以外的所有资源。使用 NotResource 时只列出不应匹配的一些资源，而不是包括将匹配的长资源列表，因此生成的策略较短。这对于在单个 Amazon 服务中应用的策略特别有用。

例如，假设您有名为 HRPayroll 的组。不应允许 HRPayroll 的成员访问除 HRBucket 存储桶中的 Amazon S3 文件夹之外的任何 Payroll 资源。以下策略显式拒绝访问除所列资源外的所有 Amazon S3 资源。但请注意，该策略不向用户授予对任何资源的访问权限。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Deny",
    "Action": "s3:*",
    "NotResource": [
      "arn:aws:s3:::HRBucket/Payroll",
      "arn:aws:s3:::HRBucket/Payroll/*"
    ]
  }
}
```

通常，为了显式拒绝对某一资源的访问，可以编写一条策略，其中使用 "Effect":"Deny"，而且包含一个单独列出每个文件夹的 Resource 元素。但是，在这种情况下，每次向 HRBucket 添加文件夹或向 Amazon S3 添加不应访问的资源时，都必须将其名称添加到 Resource 元素中的列表中。如果改用 NotResource 元素，除非将文件夹名称添加到 NotResource 元素，否则会自动拒绝用户访问新文件夹。

使用 NotResource 时应注意，在此元素中指定的资源是不受限的唯一资源。反过来，这限制了将应用于该操作的所有资源。在上述示例中，该策略只影响 Amazon S3 操作，因此只影响 Amazon S3 资源。如果 Action 元素还包含 Amazon EC2 操作，则该策略将拒绝访问 NotResource 元素中未指定的任何 EC2 资源。要了解服务中的哪些操作允许指定资源的 ARN，请参阅 Amazon 服务的操作、资源和条件键。

具有其他元素的 NotResource

您应从不将 "Effect": "Allow"、"Action": "*" 和 "NotResource": "arn:aws:s3:::HRBucket" 元素一起使用。此语句非常危险，因为它允许对在 HRBucket S3 存储桶之外的所有资源执行 Amazon 中的所有操作。这甚至允许用户向自己添加策略，从而允许他们访问 HRBucket。请勿执行此操作。

在同一语句或一个策略内的不同语句中使用 NotResource 元素和 "Effect": "Allow" 时要谨慎。NotResource 允许未显式列出的所有服务和资源，并且可能导致向用户授予超出您意图的更多权限。在同一语句中使用 NotResource 元素和 "Effect": "Deny" 将拒绝未显式列出的服务和资源。

IAM JSON 策略元素：Condition

Condition 元素（或 Condition 块）允许您指定策略生效的条件。Condition 元素是可选的。在 Condition 元素中，您可构建表达式并使用条件运算符（等于、小于和其他）将策略中的上下文键和值与请求上下文中的键和值进行匹配。要了解有关请求上下文的更多信息，请参阅请求的组成部分。

```
"Condition" : { "{condition-operator}" : { "{condition-key}" : "{condition-value}" }}
```

您在策略条件中指定的上下文键可以是全局条件上下文键或特定于服务的上下文键。全局条件上下文键具有 aws: 前缀。特定于服务的上下文键具有服务的前缀。例如，Amazon EC2 允许您使用服务独有的 ec2:InstanceType 上下文键来编写条件。要查看带 iam: 前缀的特定于服务的 IAM 上下文键，请参阅 IAM 和 Amazon STS 条件上下文密钥。

上下文键名称不区分大小写。例如，包含 aws:SourceIP 上下文键等效于测试 AWS:SourceIp。上下文键值 是否区分大小写取决于您使用的条件运算符。例如，以下条件包含 StringEquals 运算符，以确保只有 john 提出的请求才符合条件。名为 John 的用户会被拒绝访问。

```
"Condition" : { "StringEquals" : { "aws:username" : "john" }}
```

以下条件使用 StringEqualsIgnoreCase 运算符来匹配名为 john 或 John 的用户。

```
"Condition" : { "StringEqualsIgnoreCase" : { "aws:username" : "john" }}
```

某些上下文键支持允许您指定键名称的一部分的键/值对。示例包括 aws:RequestTag/*tag-key* 上下文键、Amazon KMS kms:EncryptionContext:*encryption_context_key* 和多个服务支持的 ResourceTag/*tag-key* 上下文键。

- 如果您对 Amazon EC2 等服务使用 ResourceTag/*tag-key* 上下文键，则必须指定 tag-key 的键名称。
- 键名称不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的资源标签键，但不会同时匹配两者。
- 支持这些属性的 Amazon 服务可能允许您创建多个仅大小写不同的键名称。例如，您可能需要通过 ec2=test1 和 EC2=test2 标记 Amazon EC2 实例。在使用 "aws:ResourceTag/EC2": "test1" 等条件以允许访问该资源时，键名称匹配两个标签，但仅一个值匹配。这可能会导致意外的条件失败。

> **Important**
>
> 最佳实践是，在命名键-值对属性时，请确保您的账户成员遵循一致的命名约定。示例包括标签或 Amazon KMS 加密上下文。您可以使用 aws:TagKeys 上下文键（用于标记）或 kms:EncryptionContextKeys（用于 Amazon KMS 加密上下文）强制实现此目的。

- 有关所有条件运算符及其工作原理描述的列表，请参阅条件运算符。
- 除非另行指定，否则所有上下文键均可有多个值。有关如何处理具有多个值的上下文键，请参阅 多值上下文键的集合运算符。
- 有关所有全局可用的上下文键的列表，请参阅 Amazon 全局条件上下文密钥。
- 有关每个服务定义的条件上下文键，请参阅 Amazon 服务的操作、资源和条件键。

请求上下文

当主体向 Amazon 发出请求时，Amazon 会将请求信息收集到请求上下文中。请求上下文包括有关主体、资源、操作和其他环境属性的信息。策略评估将策略中的属性与请求中发送的属性进行匹配，以评估和授权您可以在 Amazon 中执行的操作。

您可以使用 JSON 策略的 Condition 元素来针对请求上下文测试特定上下文键。例如，您可以创建一个策略，该策略使用 aws:CurrentTime 上下文键以允许用户仅在特定的日期范围内执行操作。

以下示例显示了 Martha Rivera 发送停用其 MFA 设备的请求时的请求上下文表示。

```
Principal: AROA123456789EXAMPLE
Action: iam:DeactivateMFADevice
Resource: arn:aws:iam::user/martha
Context:
  - aws:UserId=AROA123456789EXAMPLE:martha
  - aws:PrincipalAccount=1123456789012
  - aws:PrincipalOrgId=o-example
  - aws:PrincipalARN=arn:aws:iam::1123456789012:assumed-role/TestAR
  - aws:MultiFactorAuthPresent=true
  - aws:MultiFactorAuthAge=2800
  - aws:CurrentTime=...
  - aws:EpochTime=...
  - aws:SourceIp=...
```

请求上下文与允许用户删除自己的多重身份验证 (MFA) 设备的策略进行匹配，但前提是他们在过去一小时（3,600 秒）内使用 MFA 登录。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "AllowRemoveMfaOnlyIfRecentMfa",
        "Effect": "Allow",
        "Action": [
            "iam:DeactivateMFADevice"
        ],
        "Resource": "arn:aws:iam::*:user/${aws:username}",
        "Condition": {
            "NumericLessThanEquals": {"aws:MultiFactorAuthAge": "3600"}
        }
    }
}
```

在此示例中，该策略与请求上下文匹配：操作相同，资源与“*”通配符匹配，并且 aws:MultiFactorAuthAge 的值为 2800，小于 3600，因此该策略允许此授权请求。

Amazon 会评估策略中的每个上下文键，并返回 true 或 false 值。请求中不存在的上下文键会视为不匹配。

请求上下文可返回以下值：

- True - 如果请求者在过去 1 小时或更短时间内已使用 MFA 进行登录，则条件返回 true。
- False - 如果请求者在 1 小时前已使用 MFA 进行登录，则条件返回 false。
  - Not present（不存在）- 如果请求者已使用其 IAM 用户访问密钥在 Amazon CLI 或 Amazon API 中发出请求，则键不存在。在此情况下，键不存在，并且它不匹配。

> **Note**
>
> 在某些情况下，当条件键值不存在时，条件仍然可以返回 true。例如，如果您添加 ForAllValues 限定符，则当请求中不存在上下文键时，请求将返回 true。为了防止缺失的上下文键或具有空值的上下文键评估为 true，您可以在策略中包含具有 false 值的 Null 条件运算符，以检查上下文键是否存在且其值不为空。

条件块

以下示例显示 Condition 元素的基本格式：

```
"Condition": {"StringLike": {"s3:prefix": ["jane/*"]}}
```

请求中的值由上下文键表示，在此示例中为 s3:prefix。将上下文密钥值与您指定为文本值的值进行比较，例如 jane/*。要进行的比较类型由条件运算符指定（此处为 StringLike）。您可以使用典型的布尔比较（例如等于、大于和小于）来创建比较字符串、日期、数字等的条件。使用字符串运算符或 ARN 运算符时，还可以在上下文键值中使用策略变量。以下示例包括 aws:username 变量。

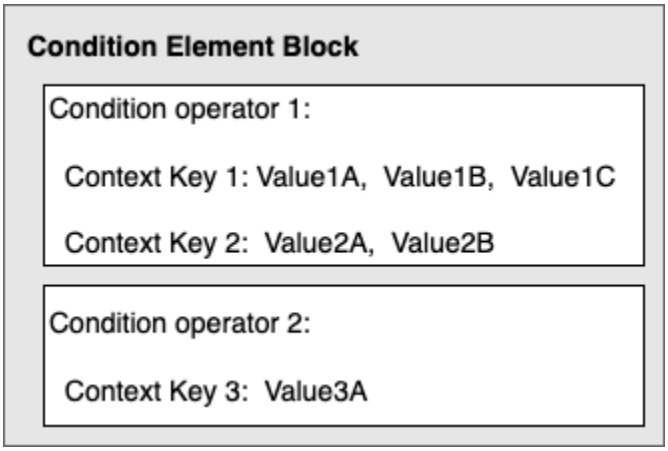
```
"Condition": {"StringLike": {"s3:prefix": ["${aws:username}/*"]}}
```

在某些情况下，上下文键可能包含多个值。例如，对 Amazon DynamoDB 的请求可能需要从表返回或更新多个属性。访问 DynamoDB 表的策略可能包含 dynamodb:Attributes 上下文键，此键包含请求中列出的所有属性。您可以使用 Condition 元素中的集合运算符，根据策略中的允许属性列表测试请求中的多个属性。有关更多信息，请参阅 多值上下文键的集合运算符。

在请求期间评估策略时，Amazon 会将键替换为请求中的相应值。(在此示例中，Amazon 将使用请求的日期和时间。) 将对条件进行评估以返回 true 或 false，然后再整体考虑策略是允许还是拒绝请求。

一个条件中包含多个值

一个 Condition 元素可以包含多个条件运算符，而每个条件运算符又可以包含多个上下文键值对。下图对此进行了说明。



有关更多信息，请参阅 多值上下文键的集合运算符。

IAM JSON 策略元素：条件运算符

在 Condition 元素中使用条件运算符来将策略中的条件键和值与请求上下文中的值进行匹配。有关 Condition 元素的更多信息，请参阅 IAM JSON 策略元素：Condition。

您可以在策略中使用的条件运算符取决于您选择的条件键。您可以选择全局条件键或特定于服务的条件键。要了解可用于全局条件键的条件运算符，请参阅Amazon 全局条件上下文密钥。要了解可用于特定于服务的条件键的条件运算符，请参阅 Amazon 服务的操作、资源和条件键并选择要查看的服务。

> **Important**
>
> 如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配，并且条件为 false。如果策略条件要求该键为不匹配，如 StringNotLike 或 ArnNotLike，并且右键不存在，则条件为 true。此逻辑适用于所有条件运算符，但 ...IfExists 和 Null check 除外。这些运算符测试请求上下文中是否存在 (exists) 键。

条件运算符可分为以下类别：

- String

- 数值
- 日期和时间
- 布尔值
- 二进制
- IP 地址
- Amazon Resource Name (ARN)（仅适用于某些服务。）
- ... IfExists（检查密钥值是否作为另一检查的一部分存在）
- 空检查（检查密钥值是否作为独立检查存在）

字符串条件运算符

利用字符串条件运算符，您可以构建基于键与字符串值的对比来限制访问的 Condition 元素。

- 策略变量 - 支持
- 通配符 - 支持

条件运算符	说明
StringEquals	精确匹配，区分大小写
StringNotEquals	否定匹配
StringEqualsIgnoreCase	精确匹配，忽略大小写
StringNotEqualsIgnoreCase	否定匹配，忽略大小写
StringLike	区分大小写的匹配。值可以在字符串中的任何位置包括多字符匹配的通配符（*）和单字符匹配的通配符（?）。您必须指定通配符才能实现部分字符串匹配。 **Note** 如果一个键包含多个值，则 StringLike 可以用集合运算符（ForAllValues:StringLike 和

条件运算符	说明
	ForAnyValue:StringLike ）来限定。有关更多信息，请参阅 多值上下文键的集合运算符。
StringNotLike	不区分大小写的无效匹配。值可以在字符串中的任何位置包括多字符匹配的通配符（*）或单字符匹配的通配符（？）。

Example字符串条件运算符

例如，以下语句包含一个 Condition 元素，该元素使用 aws:PrincipalTag 键来指定必须将发出请求的主体为 iamuser-admin 作业类别。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::111122223333:user/*",
        "Condition": {
            "StringEquals": {
                "aws:PrincipalTag/job-category": "iamuser-admin"
            }
        }
    }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。在此示例中，当主体使用附加了标签的 IAM 用户时，请求上下文中才会存在 aws:PrincipalTag/job-category 键。对于主体，如果使用附加了标签或会话标签的 IAM 角色，则也会包括它。如果没有标签的用户尝试查看或编辑访问键，则该条件返回 false，并且此语句将隐式拒绝请求。

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/job-category": "iamuser-admin" }`	`aws:PrincipalTag/job-category: - iamuser-admin`	匹配
`"StringEquals": { "aws:PrincipalTag/job-category": "iamuser-admin" }`	`aws:PrincipalTag/job-category: - dev-ops`	不匹配
`"StringEquals": { "aws:PrincipalTag/job-category": "iamuser-admin" }`	请求上下文中没有 `aws:PrincipalTag/job-category` 。	不匹配

Example使用带有字符串运算符的策略变量

以下示例使用 `StringLike` 条件运算符执行与策略变量的字符串匹配来创建策略，该策略允许 IAM 用户使用 Amazon S3 控制台管理其 Amazon S3 存储桶中的“主目录”。该策略允许对 S3 存储桶执行指定操作，前提是 `s3:prefix` 与任一指定模式相匹配。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:ListAllMyBuckets",
        "s3:GetBucketLocation"
      ],
      "Resource": "arn:aws:s3:::*"
```

```
    },
    {
      "Effect": "Allow",
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "",
            "home/",
            "home/${aws:username}/"
          ]
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "s3:*",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}",
        "arn:aws:s3:::amzn-s3-demo-bucket/home/${aws:username}/*"
      ]
    }
  ]
}
```

下表显示了 Amazon 如何根据请求上下文中的 aws:username 值，并针对不同用户评估此策略。

策略条件	请求上下文	结果
`"StringLike": { "s3:prefix": [ "home/", "home/${aws:username}/" ] }`	`aws:username: - martha_rivera`	`"StringLike": { "s3:prefix": [ "home/", "home/martha_rivera /" ] }`
`"StringLike": { "s3:prefix": [ "home/",`	`aws:username: - nikki_wolf`	`"StringLike": { "s3:prefix": [ "home/",`

策略条件	请求上下文	结果
"home/${aws:username}/"] }		"home/*nikki_wolf* /"] }
"StringLike": { "s3:prefix": ["home/", "home/${aws:username}/"] }	请求上下文中没有 aws:username 。	不匹配

有关说明如何使用 Condition 元素基于应用程序 ID 和用户 ID 限制资源的访问权限，以用于 OIDC 联合身份验证的策略示例，请参阅 Amazon S3：允许 Amazon Cognito 用户访问其存储桶中的对象。

多值字符串条件运算符

如果请求中的一个键包含多个值，则可以用集合运算符（ForAllValues 和 ForAnyValue）来限定字符串运算符。有关多个上下文键或值的评估逻辑的更多信息，请参阅 多值上下文键的集合运算符。

条件运算符	说明
ForAllValues:StringEquals ForAllValues:StringEqualsIgnoreCase	请求中条件键的所有值均须与策略中的至少一个值相匹配。
ForAnyValue:StringEquals ForAnyValue:StringEqualsIgnoreCase	请求中至少有一个条件键值须与策略中的一个值相匹配。
ForAllValues:StringNotEquals ForAllValues:StringNotEqualsIgnoreCase	否定匹配。 请求中任何上下文键值均无法与策略中的任何上下文键值相匹配。

条件运算符	说明
ForAnyValue:StringNotEquals	否定匹配。 请求中至少有一个上下文键值不得与策略中的任何上下文键值相匹配。
ForAnyValue:StringNotEqualsIgnoreCase	
ForAllValues:StringLike	请求中条件键的所有值均须与策略中的至少一个值相匹配。
ForAnyValue:StringLike	请求中至少有一个条件键值须与策略中的一个值相匹配。
ForAllValues:StringNotLike	否定匹配。 请求中任何上下文键值均无法与策略中的任何上下文键值相匹配。
ForAnyValue:StringNotLike	否定匹配。 请求中至少有一个上下文键值不得与策略中的任何上下文键值相匹配。

Example使用带有字符串条件运算符的 ForAnyValue

此示例说明如何创建允许使用 Amazon EC2 CreateTags 操作将标签附加到实例的基于身份的策略。在使用 StringEqualsIgnoreCase 的情况下，只有在标签包含值为 preprod 或 storage 的 environment 键时，才能附加标签。向运算符追加 IgnoreCase 时，允许任何现有标签值的大写（例如 preprod、Preprod 和 PreProd）解析为 true。

当添加带有 aws:TagKeys 条件键的 ForAnyValue 修饰符时，请求中至少有一个标签键值须与 environment 值相匹配。ForAnyValue 比较区分大小写，这可以防止用户为标签键使用不正确的大小写，例如使用 Environment 而不是 environment。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
```

```
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2:*:*:instance/*",
    "Condition": {
      "StringEqualsIgnoreCase": {
        "aws:RequestTag/environment": [
          "preprod",
          "storage"
        ]
      },
      "ForAnyValue:StringEquals": {
        "aws:TagKeys": "environment"
      }
    }
  }
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringEqualsIgnoreCase": { "aws:RequestTag/environment": [ "preprod", "storage" ] }, "ForAnyValue:StringEquals": { "aws:TagKeys": "environment" }`	`aws:TagKeys: - environment aws:RequestTag/environment: - preprod`	匹配
`"StringEqualsIgnoreCase": { "aws:RequestTag/environment": [ "preprod", "storage"`	`aws:TagKeys: - environment - costcenter aws:RequestTag/environment:`	匹配

策略条件	请求上下文	结果

]
},
"ForAnyValue:StringEqu
als": {
"aws:TagKeys":
"environment"
}
	- PreProd	

"StringEqualsIgnor
eCase": {
"aws:RequestTag/en
vironment": [
"preprod",
"storage"
]
},
"ForAnyValue:StringEqu
als": {
"aws:TagKeys":
"environment"
}
	aws:TagKeys:
	- Environment
	aws:RequestTag/
	Environment:
	- preprod
		不匹配

"StringEqualsIgnor
eCase": {
"aws:RequestTag/en
vironment": [
"preprod",
"storage"
]
},
"ForAnyValue:StringEqu
als": {
"aws:TagKeys":
"environment"
}
	aws:TagKeys:
	- costcenter
	aws:RequestTag/e
	nvironment:
	- preprod
		不匹配

策略条件	请求上下文	结果
`"StringEqualsIgnoreCase": { "aws:RequestTag/environment": [ "preprod", "storage" ] }, "ForAnyValue:StringEquals": { "aws:TagKeys": "environment" }`	请求上下文中没有 aws:TagKeys 。 `aws:RequestTag/environment: - storage`	不匹配
`"StringEqualsIgnoreCase": { "aws:RequestTag/environment": [ "preprod", "storage" ] }, "ForAnyValue:StringEquals": { "aws:TagKeys": "environment" }`	`aws:TagKeys: - environment` 请求上下文中没有 aws:RequestTag/environment 。	不匹配

策略条件	请求上下文	结果
`"StringEqualsIgnoreCase": { "aws:RequestTag/environment": [ "preprod", "storage" ] }, "ForAnyValue:StringEquals": { "aws:TagKeys": "environment" }`	请求上下文中没有 `aws:TagKeys` 。 请求上下文中没有 `aws:RequestTag/environment` 。	不匹配

通配符匹配

字符串条件运算符执行不强制使用预定义格式的无模式匹配。ARN 和 Date 条件运算符是字符串运算符的子集，用于强制执行条件键值的结构。

我们建议您使用与要比较键的值对应的条件运算符。例如，在将键与字符串值进行比较时，应使用the section called “字符串条件运算符”。同样，在将键与 ARN 值进行比较时，应使用 the section called “Amazon Resource Name (ARN) 条件运算符”。

Example

此示例演示了如何为组织的资源创建边界。该策略中的条件拒绝访问 Amazon S3 操作，除非当前访问的资源位于 Amazon Organizations 中的特定组织单位 (OU) 中。Amazon Organizations 路径是 Organizations 实体结构的文本表示形式。

条件的要求是 `aws:ResourceOrgPaths` 包含列出的任何 OU 路径。由于 `aws:ResourceOrgPaths` 为多值条件，因此该策略使用 `ForAllValues:StringNotLike` 运算符来比较 `aws:ResourceOrgPaths` 的值和策略中的 OU 列表。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "DenyS3AccessOutsideMyBoundary",
      "Effect": "Deny",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "ForAllValues:StringNotLike": {
          "aws:ResourceOrgPaths": [
            "o-acorg/r-acroot/ou-acroot-mediaou/",
            "o-acorg/r-acroot/ou-acroot-sportsou/*"
          ]
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"ForAllValues:StringNotLike": { "aws:ResourceOrgPaths": [ "o-acorg/r-acroot/ou-acroot-mediaou/", "o-acorg/r-acroot/ou-acroot-sportsou/*" ] }`	`aws:ResourceOrgPaths: - o-acorg/r-acroot/ou-acroot-sportsou/costcenter/`	匹配
`"ForAllValues:StringNotLike": { "aws:ResourceOrgPaths": [ "o-acorg/r-acroot/ou-acroot-mediaou/",`	`aws:ResourceOrgPaths: - o-acorg/r-acroot/ou-acroot-mediaou/costcenter/`	不匹配

策略条件	请求上下文	结果
"o-acorg/r-acroot/ou-acroot-sportsou/*"] }		
"ForAllValues:StringNotLike": { "aws:ResourceOrgPaths": ["o-acorg/r-acroot/ou-acroot-mediaou/", "o-acorg/r-acroot/ou-acroot-sportsou/*"] }	请求中没有 aws:ResourceOrgPaths: 。	不匹配

数字条件运算符

利用数字条件运算符，您可以构建基于键与整数或小数值的对比来限制访问的 Condition 元素。

- 策略变量 – 不支持
- 通配符 - 不支持

条件运算符	说明
NumericEquals	Matching
NumericNotEquals	否定匹配
NumericLessThan	“小于”匹配
NumericLessThanEquals	“小于或等于”匹配
NumericGreaterThan	“大于”匹配

条件运算符	说明
NumericGreaterThanEquals	“大于或等于”匹配

例如，以下语句包含一个 Condition 元素，该元素使用 NumericLessThanEquals 条件运算符与 s3:max-keys 密钥来指定请求者一次最多可在 amzn-s3-demo-bucket 内列出 10 个对象。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "s3:ListBucket",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
    "Condition": {"NumericLessThanEquals": {"s3:max-keys": "10"}}
  }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。在此示例中，当您执行 ListBucket 操作时，s3:max-keys 键始终存在于请求中。如果此策略允许所有 Amazon S3 操作，则只允许包含值小于或等于 10 的 max-keys 上下文密钥的操作。

日期条件运算符

利用日期条件运算符，您可以构建基于键与日期/时间值的对比来限制访问的 Condition 元素。您可以将这些条件运算符与 aws:CurrentTime 键或 aws:EpochTime 键配合使用。您必须指定日期/时间值，且其中一个 W3C 实现要采用 ISO 8601 日期格式或新纪元 (UNIX) 时间格式。

- 策略变量 – 不支持
- 通配符 - 不支持

条件运算符	说明
DateEquals	匹配特定日期

条件运算符	说明
DateNotEquals	否定匹配
DateLessThan	在特定日期和时间之前匹配。
DateLessThanEquals	在特定日期和时间或之前匹配
DateGreaterThan	在特定日期和时间之后匹配
DateGreaterThanEquals	在特定日期和时间或之后匹配

例如，以下语句包含一个 Condition 元素，该元素使用 DateGreaterThan 条件运算符与 aws:TokenIssueTime 键。此条件指定用于发出请求的临时安全凭证在 2020 年签发。此策略可以通过编程方式每天更新，以确保账户成员使用最新的凭证。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::111122223333:user/*",
        "Condition": {
            "DateGreaterThan": {
                "aws:TokenIssueTime": "2020-01-01T00:00:01Z"
            }
        }
    }
}
```

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。只有在主体使用临时凭证发出请求时，才会在请求上下文中包含 aws:TokenIssueTime 键。使用访问密钥发出的 Amazon CLI、Amazon API 或 Amazon 开发工具包请求中未提供此键。在此示例中，如果 IAM 用户尝试查看或编辑访问键，请求将被拒绝。

布尔值条件运算符

利用布尔值条件，您可以构建基于键与 `true` 或 `false` 的对比来限制访问的 `Condition` 元素。

如果一个键包含多个值，则可以用集合运算符（`ForAllValues` 和 `ForAnyValue`）来限定布尔运算符。有关多个上下文键或值的评估逻辑的更多信息，请参阅 多值上下文键的集合运算符。

- 策略变量 - 支持
- 通配符 - 不支持

条件运算符	说明
`Bool`	布尔值匹配
`ForAllValues:Bool`	与布尔数组数据类型一起使用。上下文键值中所有布尔值均须与策略中的布尔值相匹配。 为防止 `ForAllValues` 运算符将缺失的上下文键或带有空值的上下文键评估为“允许”，可以在策略中包含 Null 条件运算符。
`ForAnyValue:Bool`	与布尔数组数据类型一起使用。上下文键值中至少有一个布尔值须与策略中的布尔值相匹配。

Example布尔值条件运算符

以下基于身份的策略使用 `Bool` 条件运算符搭配 `aws:SecureTransport` 键拒绝将对象和对象标签复制到目标存储桶及其内容（如果请求不是通过 SSL 进行的）。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "BooleanExample",
      "Action": "s3:ReplicateObject",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket",
        "arn:aws:s3:::amzn-s3-demo-bucket/*"
      ],
      "Condition": {
        "Bool": {
          "aws:SecureTransport": "false"
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"Bool": { "aws:SecureTranspo rt": "false" }`	`aws:SecureTransport: - false`	匹配
`"Bool": { "aws:SecureTranspo rt": "false" }`	`aws:SecureTransport: - true`	不匹配
`"Bool": { "aws:SecureTranspo rt": "false" }`	请求上下文中没有 `aws:Secur eTransport` 。	不匹配

二进制条件运算符

利用 BinaryEquals 条件运算符，您可以构建测试二进制格式键值的 Condition 元素。它会比较指定密钥字节的值和策略中 base-64 编码表示的二进制值。如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。

- 策略变量 – 不支持
- 通配符 - 不支持

```
"Condition" : {
  "BinaryEquals": {
    "key" : "QmluYXJ5VmFsdWVJbkJhc2U2NA=="
  }
}
```

策略条件	请求上下文	结果
`"BinaryEquals": { "key" : "QmluYXJ5 VmFsdWVJbkJhc2U2NA==" }`	`key: - QmluYXJ5VmFsdWVJbk Jhc2U2NA==`	匹配
`"BinaryEquals": { "key" : "QmluYXJ5 VmFsdWVJbkJhc2U2NA==" }`	`key: - ASIAIOSFODNN7EXAMP LE`	不匹配
`"BinaryEquals": { "key" : "QmluYXJ5 VmFsdWVJbkJhc2U2NA==" }`	请求上下文中没有 key。	不匹配

IP 地址条件运算符

利用 IP 地址条件运算符，您可以构建 Condition 元素，它们会基于键与 IPv4 或 IPv6 地址或 IP 地址范围的对比来限制访问。可以通过 aws:SourceIp 键来使用它们。该值必须采用标准的 CIDR 格式

(例如 203.0.113.0/24 或 2001:DB8:1234:5678::/64)。如果您指定的 IP 地址没有关联的路由前缀，IAM 将使用 /32 作为默认前缀值。

某些 Amazon 服务支持 IPv6，使用 :: 表示一系列 0。要了解某项服务是否支持 IPv6，请参阅该服务的文档。

- 策略变量 – 不支持
- 通配符 - 不支持

条件运算符	说明
IpAddress	指定的 IP 地址或范围
NotIpAddress	除指定 IP 地址或范围外的所有 IP 地址

Example IP 地址条件运算符

下列语句使用 IpAddress 条件运算符与 aws:SourceIp 键来指定必须从 IP 范围 203.0.113.0 至 203.0.113.255 发出请求。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Effect": "Allow",
        "Action": "iam:*AccessKey*",
        "Resource": "arn:aws:iam::111122223333:user/*",
        "Condition": {
            "IpAddress": {
                "aws:SourceIp": "203.0.113.0/24"
            }
        }
    }
}
```

aws:SourceIp 条件键解析为发出请求的 IP 地址。如果请求源自 Amazon EC2 实例，则 aws:SourceIp 计算为实例的公有 IP 地址。

如果请求上下文中没有您在策略条件中指定的键，则这些值不匹配。aws:SourceIp 键始终存在于请求上下文中，但请求者使用 VPC 终端节点发出请求时除外。在这种情况下，条件返回 false，并且此语句隐式拒绝请求。

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"IpAddress": { "aws:SourceIp": "203.0.113.0/24" }`	`aws:SourceIp: - 203.0.113.1`	匹配
`"IpAddress": { "aws:SourceIp": "203.0.113.0/24" }`	`aws:SourceIp: - 198.51.100.1`	不匹配

下面的示例说明如何结合使用 IPv4 和 IPv6 地址来覆盖组织的所有有效 IP 地址。建议您使用您的 IPv6 地址范围以及已拥有的 IPv4 范围来更新组织的策略，以确保策略在您过渡到 IPv6 时继续有效。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "someservice:*",
    "Resource": "*",
    "Condition": {
      "IpAddress": {
        "aws:SourceIp": [
          "203.0.113.0/24",
          "2001:DB8:1234:5678::/64"
        ]
      }
    }
  }
```

```
}
```

如果以用户身份直接调用测试的 API，aws:SourceIp 条件键仅在 JSON 策略中有效。如果改为使用服务代表您调用目标服务，则目标服务看到的是进行调用的服务的 IP 地址而不是源用户的 IP 地址。举例来说，如果使用 Amazon CloudFormation 调用 Amazon EC2 来构建实例，则会发生这种情况。目前，无法通过进行调用的服务将源 IP 地址传递给目标服务以在 JSON 策略中进行评估。对于这些服务 API 调用类型，请勿使用 aws:SourceIp 条件键。

Amazon Resource Name (ARN) 条件运算符

利用 Amazon Resource Name (ARN) 条件运算符，您可以构建基于键与 ARN 的对比来限制访问的 Condition 元素。ARN 被视为一个字符串。

- 策略变量 - 支持
- 通配符 - 支持

条件运算符	说明
ArnEquals ，ArnLike	区分大小写的 ARN 匹配。ARN 的六个由冒号分隔开的部分都要单独检查，每一个部分都可包括一个多字符匹配通配符（*）或一个单字符匹配通配符（？）。ArnEquals 和 ArnLike 条件运算符的行为相同。
ArnNotEquals ，ArnNotLike	ARN 无效匹配。ArnNotEquals 和 ArnNotLike 条件运算符的行为相同。

Example ARN 条件运算符

以下基于资源的策略示例显示附加到 Amazon SQS 队列（您希望向其发送 SNS 消息的队列）的策略。它授予 Amazon SNS 将消息发送到所选队列的权限，但仅在该服务代表特定 Amazon SNS 主题发送这些消息时才授予此权限。您在 Resource 字段中指定队列，Amazon SNS 主题则作为 SourceArn 密钥的值。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
```

```
        "Effect": "Allow",
        "Principal": {
            "Service": "sns.amazonaws.com"
        },
        "Action": "SQS:SendMessage",
        "Resource": "arn:aws:sqs:us-east-1:123456789012:QUEUE-ID",
        "Condition": {
            "ArnEquals": {
                "aws:SourceArn": "arn:aws:sns:us-east-1:123456789012:TOPIC-ID"
            }
        }
    }
}
```

仅在资源触发一项服务以代表资源拥有者调用另一项服务时，aws:SourceArn 键存在于请求上下文中。如果 IAM 用户尝试直接执行此操作，则条件返回 false，并且此语句隐式拒绝请求。

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"ArnEquals": { "aws:SourceArn": "arn:aws:sns:us-west-2:123456789012:TOPIC-ID" }`	`aws:SourceArn: - arn:aws:sns:us-west-2:123456789012:TOPIC-ID`	匹配
`"ArnEquals": { "aws:SourceArn": "arn:aws:sns:us-west-2:123456789012:TOPIC-ID" }`	`aws:SourceArn: - arn:aws:sns:us-west-2:777788889999:TOPIC-ID`	不匹配
`"ArnEquals": { "aws:SourceArn": "arn:aws:sns:us-west-2:123456789012:TOPIC-ID"`	请求上下文中没有 aws:SourceArn 。	不匹配

策略条件	请求上下文	结果
}		

多值 ARN 条件运算符

如果请求中的一个键包含多个值，则可以用集合运算符（ForAllValues 和 ForAnyValue）来限定 ARN 运算符。有关多个上下文键或值的评估逻辑的更多信息，请参阅 多值上下文键的集合运算符。

条件运算符	说明
ForAllValues:ArnEquals ForAllValues:ArnLike	请求上下文中所有 ARN 均须与策略中的至少一个 ARN 模式相匹配。
ForAnyValue:ArnEquals ForAnyValue:ArnLike	请求上下文中至少有一个 ARN 须与策略中的一个 ARN 模式相匹配。
ForAllValues:ArnNotEquals ForAllValues:ArnNotLike	否定匹配。 请求上下文中任何 ARN 都无法与策略中的任何字符串 ARN 模式匹配。
ForAnyValue:ArnNotEquals ForAnyValue:ArnNotLike	否定匹配。 请求上下文中至少有一个 ARN 不得与策略中的任何 ARN 模式相匹配。

Example使用带有 ARN 条件运算符的 ForAllValues

以下示例使用 ForAllValues:ArnLike 创建或更新 Amazon CloudWatch Logs 日志的逻辑传输源。条件块中包含条件键 logs:LogGeneratingResourceArns，可筛选请求中传递的日志生成资源 ARN。使用此条件运算符，请求中的所有 ARN 均须与策略中的至少一个 ARN 相匹配。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": "*",
            "Action": "logs:PutDeliverySource",
            "Resource": "arn:aws:logs:us-east-1:123456789012:delivery-source:*",
            "Condition": {
                "ForAllValues:ArnLike": {
                    "logs:LogGeneratingResourceArns": [
                        "arn:aws:cloudfront::123456789012:distribution/*",
                        "arn:aws:cloudfront::123456789012:distribution/support*"
                    ]
                }
            }
        }
    ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"ForAllValues:ArnLike": { "logs:LogGeneratingResourceArns": [ "arn:aws::cloudfront:123456789012:distribution/*", "arn:aws::cloudfront:123456789012:distribution/support*" ] }`	`logs:LogGeneratingResourceArns: - arn:aws::cloudfront:123456789012:distribution/costcenter`	匹配
`"ForAllValues:ArnLike": { "logs:LogGeneratingResourceArns": [`	`logs:LogGeneratingResourceArns: - arn:aws::cloudfront:123456789012:distribution/costcenter`	匹配

策略条件	请求上下文	结果
"arn:aws::cloudfront:123456789012:distribution/*", "arn:aws::cloudfront:123456789012:distribution/support*"] }	- arn:aws::cloudfront:123456789012:distribution/support2025	

"ForAllValues:ArnLike": {
"logs:LogGeneratingResourceArns": [
"arn:aws::cloudfront:123456789012:distribution/*",
"arn:aws::cloudfront:123456789012:distribution/support*"
]
}
	logs:LogGeneratingResourceArns:
	- arn:aws::cloudfront:123456789012:distribution/costcenter
	- arn:aws::cloudfront:123456789012:distribution/admin
		不匹配

"ForAllValues:ArnLike": {
"logs:LogGeneratingResourceArns": [
"arn:aws::cloudfront:123456789012:distribution/*",
"arn:aws::cloudfront:123456789012:distribution/support*"
]
}
	logs:LogGeneratingResourceArns:
	- arn:aws::cloudfront:777788889999:distribution/costcenter
		不匹配

策略条件	请求上下文	结果
`"ForAllValues:ArnLike": { "logs:LogGeneratingResourceArns": [ "arn:aws::cloudfront:123456789012:distribution/*", "arn:aws::cloudfront:123456789012:distribution/support*" ] }`	请求上下文中没有 `logs:LogGeneratingResourceArns` 。	匹配

如果请求中没有上下文键或者上下文键值解析为空数据集（如空字符串），则 `ForAllValues` 限定符也会返回 true。为了防止缺失的上下文键或具有空值的上下文键评估为 true，您可以在策略中包含具有 `false` 值的 Null 条件运算符，以检查上下文键是否存在且其值不为空。

...IfExists 条件运算符

除 `Null` 条件外，您可在任何条件运算符名称的末尾添加 `IfExists`，例如，`StringLikeIfExists`。如果您是指“如果请求的内容中存在条件键，则依照策略所述来处理键。如果该键不存在，则条件元素的计算结果将为 true。”语句中其他条件因素仍然可以导致不匹配，但使用 `...IfExists` 检查时没有缺失键。如果您使用带有否定条件运算符（如 `StringNotEqualsIfExists`）的 `"Effect": "Deny"` 元素，则即使条件键不存在，请求仍会被拒绝。

使用 **`IfExists`** 的示例

许多条件键描述有关特定类型的资源的信息，仅当访问该类型的资源时才存在。这些条件键在其他类型的资源上不存在。当策略语句仅适用于一种类型的资源时，这不会导致问题。但是，有时单个语句可以适用于多种类型的资源，例如当策略语句从多个服务引用操作时，或是当服务中的给定操作访问同一服务中的多种不同资源类型时。在这种情况下，在策略语句中包含仅适用于一种资源的条件键可能会导致策略语句中的 `Condition` 元素失败，从而使语句的 `"Effect"` 不适用。

例如，请考虑以下策略示例：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Sid": "THISPOLICYDOESNOTWORK",
    "Effect": "Allow",
    "Action": "ec2:RunInstances",
    "Resource": "*",
    "Condition": {"StringLike": {"ec2:InstanceType": [
      "t1.*",
      "t2.*",
      "m3.*"
    ]}}
  }
}
```

上述策略的意图是使用户能够启动类型为 t1、t2 或 m3 的任何实例。但是，启动实例要求访问除了实例本身之外的许多资源；例如映像、密钥对、安全组等。会针对启动实例所需的每个资源来评估整个语句。这些其他资源没有 ec2:InstanceType 条件键，因此 StringLike 检查会失败，并且不会向用户授予启动任何 实例类型的能力。

要解决此问题，请改用 StringLikeIfExists 条件运算符。这样，仅当条件键存在时才会进行测试。您会读到以下策略内容：If the resource being checked has an "ec2:InstanceType" condition key, then allow the action only if the key value begins with t1., t2., or m3.. 如果所检查的资源没有改条件键，则无需担心。” 条件键值中的星号 (*) 与 StringLikeIfExists 条件运算符一起使用时，会被解释为通配符以实现部分字符串匹配。DescribeActions 语句包含在控制台中查看实例所需的操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "RunInstance",
      "Effect": "Allow",
      "Action": "ec2:RunInstances",
      "Resource": "*",
```

```
      "Condition": {
        "StringLikeIfExists": {
          "ec2:InstanceType": [
            "t1.*",
            "t2.*",
            "m3.*"
          ]
        }
      }
    },
    {
      "Sid": "DescribeActions",
      "Effect": "Allow",
      "Action": [
        "ec2:DescribeImages",
        "ec2:DescribeInstances",
        "ec2:DescribeVpcs",
        "ec2:DescribeKeyPairs",
        "ec2:DescribeSubnets",
        "ec2:DescribeSecurityGroups"
      ],
      "Resource": "*"
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringLikeIfExists": { "ec2:InstanceType": [ "t1.*", "t2.*", "m3.*" ] }`	`ec2:InstanceType: - t1.micro`	匹配

策略条件	请求上下文	结果
`"StringLikeIfExists": { "ec2:InstanceType": [ "t1.*", "t2.*", "m3.*" ] }`	`ec2:InstanceType: - m2.micro`	不匹配
`"StringLikeIfExists": { "ec2:InstanceType": [ "t1.*", "t2.*", "m3.*" ] }`	请求上下文中没有 `ec2:InstanceType` 。	匹配

用于检查条件键是否存在的条件运算符

使用 `Null` 条件运算符检查授权时是否缺少条件键。在策略语句中使用 `true`（密钥不存在 - 为 null）或 `false`（密钥存在且值不为 null）。

您不能对策略变量使用 `Null` 条件运算符。

例如，您可以使用此条件运算符来确定用户是使用临时凭证还是自己的凭证发出请求。如果用户使用的是临时凭证，则键 `aws:TokenIssueTime` 存在并具有一个值。以下示例显示了一个条件，该条件规定用户必须使用临时凭证（键不能缺失）才能使用 Amazon EC2 API。

JSON

```
{
  "Version":"2012-10-17",
  "Statement":{
      "Action":"ec2:*",
```

```
      "Effect":"Allow",
      "Resource":"*",
      "Condition":{"Null":{"aws:TokenIssueTime":"false"}}
  }
}
```

具有多个上下文键或值的条件

您可以使用策略的 Condition 元素测试请求中的多个上下文键或单个上下文键的多个值。在以编程方式或通过 Amazon向 Amazon Web Services 管理控制台 发出请求时，请求包含有关您的主体、操作和标签等的信息。您可以使用上下文键测试请求中匹配的上下文键的值，并在策略条件中指定上下文键。要了解请求中包含的信息和数据，请参阅请求上下文。

主题

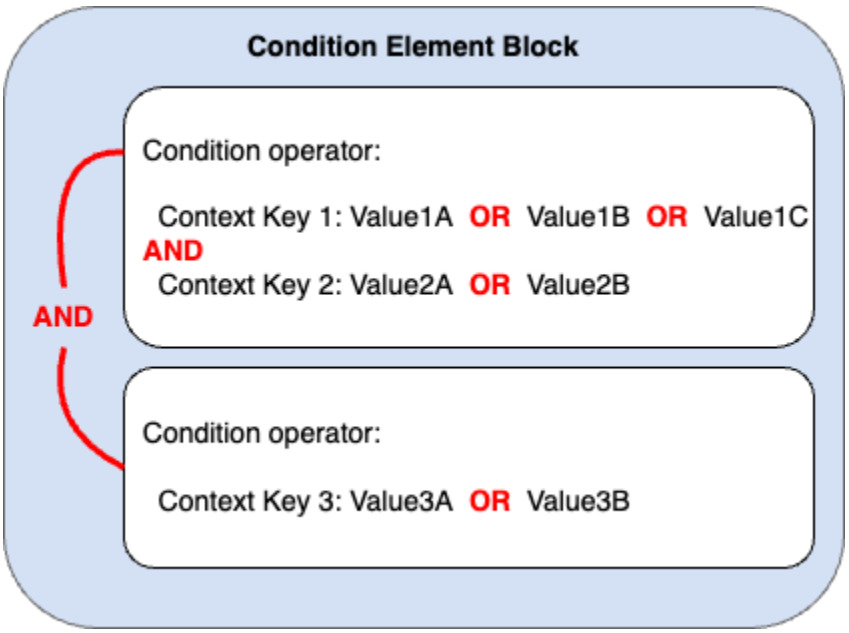
- 多个上下文键或值的评估逻辑
- 否定匹配条件运算符的评估逻辑

多个上下文键或值的评估逻辑

一个 Condition 元素可以包含多个条件运算符，而每个条件运算符又可以包含多个上下文键值对。除非另行指定，否则大多数上下文键都支持使用多个值。

- 如果您的策略语句具有多个条件运算符，则使用逻辑 AND 评估条件运算符。
- 如果您的策略语句将多个上下文键附加到单个条件运算符，则使用逻辑 AND 评估上下文键。
- 如果单个条件运算符包含一个上下文键的多个值，则使用逻辑 OR 评估这些值。
- 如果单个否定匹配条件运算符包含一个上下文键的多个值，则使用逻辑 NOR 评估这些值。

条件元素块中的所有上下文键都必须解析为 true 才能调用所需的 Allow 或 Deny 效果。下图说明了具有多个条件运算符和上下文键值对的条件的评估逻辑。



例如，以下 S3 存储桶策略说明了上图在策略中的表示方式。此条件块包含条件运算符 `StringEquals` 和 `ArnLike`，以及上下文键 `aws:PrincipalTag` 和 `aws:PrincipalArn`。要调用所需的 `Allow` 或 Deny 效果，条件元素块中的所有上下文键都必须解析为 true。提出请求的用户必须同时拥有 department 和 role 这两个主体标签键，包括策略中指定的一个标签键值。此外，发出请求的用户的主体 ARN 必须与策略中指定的 `aws:PrincipalArn` 值之一匹配才能评估为 true。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
            "hr",
            "legal"
```

```
          ],
          "aws:PrincipalTag/role": [
            "audit",
            "security"
          ]
        },
        "ArnLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana",`	`aws:PrincipalTag/department: legal aws:PrincipalTag/role: audit aws:PrincipalArn: arn:aws:iam::222222222222:user/Mary`	匹配

策略条件	请求上下文	结果
"arn:aws:iam::222222222222:user/Mary"] }		

"StringEquals": {
"aws:PrincipalTag/department": [
"finance",
"hr",
"legal"
],
"aws:PrincipalTag/role": [
"audit",
"security"
]
},
"ArnLike": {
"aws:PrincipalArn": [
"arn:aws:iam::222222222222:user/Ana",
"arn:aws:iam::222222222222:user/Mary"
]
}
	aws:PrincipalTag/department: hr
	aws:PrincipalTag/role: audit
	aws:PrincipalArn: arn:aws:iam::222222222222:user/ *Nikki*
		不匹配

策略条件	请求上下文	结果

"StringEquals": {
  "aws:PrincipalTag/department": [
  "finance",
  "hr",
  "legal"
  ],
  "aws:PrincipalTag/role": [
  "audit",
  "security"
  ]
},
"ArnLike": {
  "aws:PrincipalArn": [
  "arn:aws:iam::222222222222:user/Ana",
  "arn:aws:iam::222222222222:user/Mary"
  ]
}
	aws:PrincipalTag/department: hr
	aws:PrincipalTag/role: *payroll*
	aws:PrincipalArn:
	  arn:aws:iam::222222222222:user/Mary
		不匹配

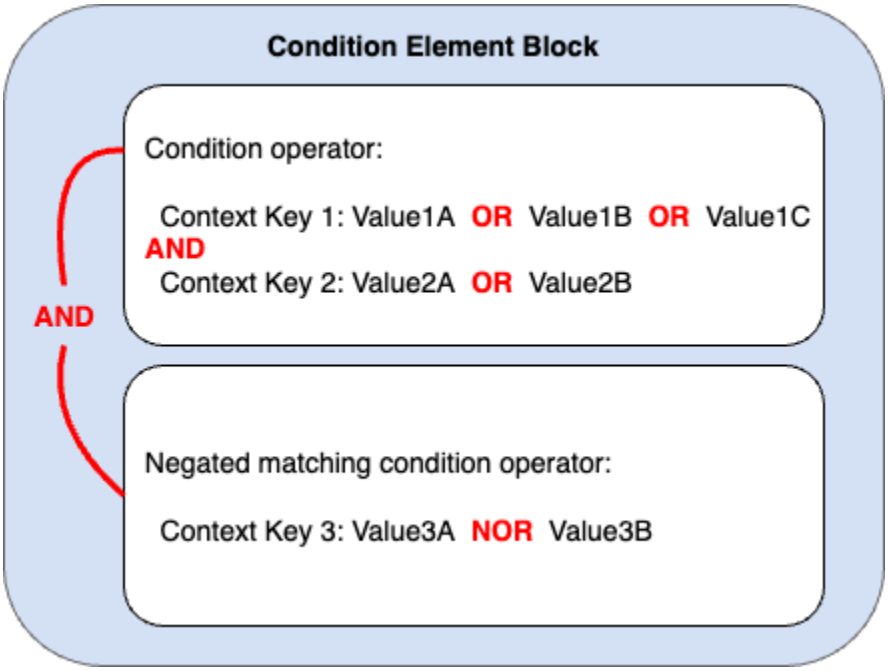
策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana", "arn:aws:iam::222222222222:user/Mary" ] }`	请求上下文中没有 aws:PrincipalTag/role 。 `aws:PrincipalTag/department: hr aws:PrincipalArn: arn:aws:iam::222222222222:user/Mary`	不匹配

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana", "arn:aws:iam::222222222222:user/Mary" ] }`	请求上下文中没有 aws:PrincipalTag 。 `aws:PrincipalArn: arn:aws:iam::22222222222222:user/Mary`	不匹配

否定匹配条件运算符的评估逻辑

一些条件运算符（如 StringNotEquals 或 ArnNotLike），使用否定匹配将策略中的上下文键值对与请求中的上下文键值对进行比较。当使用否定匹配条件运算符为策略中的单个上下文键指定多个值时，有效权限的工作方式类似于逻辑 NOR。在否定匹配中，仅当所有值都计算为 false 时，逻辑 NOR 或 NOT OR 才会返回 true。

下图说明了具有多个条件运算符和上下文键值对的条件的评估逻辑。此图包括上下文键 3 的否定匹配条件运算符。



例如，以下 S3 存储桶策略说明了上图在策略中的表示方式。此条件块包含条件运算符 `StringEquals` 和 `ArnNotLike`，以及上下文键 `aws:PrincipalTag` 和 `aws:PrincipalArn`。要调用所需的 `Allow` 或 `Deny` 效果，条件元素块中的所有上下文键都必须解析为 true。提出请求的用户必须同时拥有 department 和 role 这两个主体标签键，包括策略中指定的一个标签键值。由于 `ArnNotLike` 条件运算符使用否定匹配，发出请求的用户的主体 ARN 不得与策略中指定的任何 `aws:PrincipalArn` 值匹配才能评估为 true。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExamplePolicy",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::222222222222:root"
      },
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/department": [
            "finance",
```

```
            "hr",
            "legal"
          ],
          "aws:PrincipalTag/role": [
            "audit",
            "security"
          ]
        },
        "ArnNotLike": {
          "aws:PrincipalArn": [
            "arn:aws:iam::222222222222:user/Ana",
            "arn:aws:iam::222222222222:user/Mary"
          ]
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnNotLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana",`	`aws:PrincipalTag/department: legal aws:PrincipalTag/role: audit aws:PrincipalArn: arn:aws:iam::222222222222:user/Nikki`	匹配

策略条件	请求上下文	结果
"arn:aws:iam::222222222222:user/Mary"] }		

"StringEquals": {
"aws:PrincipalTag/department": [
"finance",
"hr",
"legal"
],
"aws:PrincipalTag/role": [
"audit",
"security"
]
},
"ArnNotLike": {
"aws:PrincipalArn": [
"arn:aws:iam::222222222222:user/Ana",
"arn:aws:iam::222222222222:user/Mary"
]
}
	aws:PrincipalTag/department: hr
	aws:PrincipalTag/role: audit
	aws:PrincipalArn: arn:aws:iam::222222222222:user/ *Mary*
		不匹配

策略条件	请求上下文	结果

"StringEquals": {
  "aws:PrincipalTag/department": [
  "finance",
  "hr",
  "legal"
  ],
  "aws:PrincipalTag/role": [
  "audit",
  "security"
  ]
},
"ArnNotLike": {
  "aws:PrincipalArn": [
  "arn:aws:iam::222222222222:user/Ana",
  "arn:aws:iam::222222222222:user/Mary"
  ]
}
	aws:PrincipalTag/department: hr
	aws:PrincipalTag/role: *payroll*
	aws:PrincipalArn:
	  arn:aws:iam::22222222222222:user/Nikki
		不匹配

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnNotLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana", "arn:aws:iam::222222222222:user/Mary" ] }`	> 请求上下文中没有 aws:PrincipalTag/role 。 `aws:PrincipalTag/department: hr aws:PrincipalArn: arn:aws:iam::22222222222222:user/Nikki`	不匹配

策略条件	请求上下文	结果
`"StringEquals": { "aws:PrincipalTag/department": [ "finance", "hr", "legal" ], "aws:PrincipalTag/role": [ "audit", "security" ] }, "ArnNotLike": { "aws:PrincipalArn": [ "arn:aws:iam::222222222222:user/Ana", "arn:aws:iam::222222222222:user/Mary" ] }`	请求上下文中没有 aws:PrincipalTag 。 `aws:PrincipalArn: arn:aws:iam::22222222222222:user/Nikki`	不匹配

单值和多值上下文键

单值键和多值上下文键之间的差异取决于请求上下文中的值数量，而不是策略条件中的值数量。

- 单值上下文键在请求上下文中最多有一个值。例如，当您在 Amazon 中标记资源时，每个资源标签都存储为键值对。由于资源标签键只能具有单个标签值，因此 the section called “ResourceTag” 为单值上下文键。请勿将条件集合运算符用于单值上下文键。
- 多值上下文键在请求上下文中可以有多个值。例如，当您在 Amazon 中标记资源时，您可以在请求中包含多个标签键值对。因此，the section called “TagKeys” 是多值上下文键。多值上下文键需要条件条件集合运算符。

例如，一个请求最多可以来自一个 VPC 端点，因此 the section called “SourceVpce” 是单值上下文键。由于服务可以有多个属于该服务的服务主体名称，因此 the section called “PrincipalServiceNamesList” 是多值上下文键。

> **Important**
>
> 单值键和多值上下文键之间的差异取决于请求上下文中的值数量，而不是策略条件中的值数量。

关键点

- 单值和多值分类作为值类型包含在 Amazon 全局条件上下文密钥 主题每个条件上下文键的描述中。
- 服务授权参考中的多值上下文键使用 `ArrayOf` 前缀，后跟条件运算符类别类型，例如 `ArrayOfString` 或 `ArrayOfARN`，表示请求可能包含条件上下文键的多个值。
- 您可以使用任何可用的单值上下文键作为策略变量，但不能使用多值上下文键作为策略变量。有关策略变量的更多信息，请参阅 IAM policy 元素：变量和标签。
- 使用包含键值对的上下文键时，请务必注意，尽管可以有多个标签键值，但每个 `tag-key` 只能有一个值。
  - the section called “PrincipalTag”、the section called “RequestTag” 和 the section called “ResourceTag” 是单值上下文键。
  - the section called “TagKeys” 定义请求中允许使用的标签键，但不包含标签键值。由于您可以在一个请求中包含多个标签键值对，因此 `aws:TagKeys` 是一个多值上下文键。
- 多值上下文键需要条件条件集合运算符。请勿将条件集合运算符 `ForAllValues` 或 `ForAnyValue` 用于单值上下文键。使用带有单值上下文键的条件集合运算符可能会导致过于宽容的策略。

多值上下文键的集合运算符

要将条件上下文键与具有多个键值的请求上下文键进行比较，必须使用 `ForAllValues` 或 `ForAnyValue` 集合运算符。这些集合运算符用于比较两组值，例如请求中的标签集和策略条件中的标签集。

限定词 `ForAllValues` 和 `ForAnyValue` 为条件运算符添加了集合运算功能，从而使您可以针对策略条件中的多个上下文键值测试具有多个值的请求上下文键。此外，如果在策略中包含带有通配符或变量的多值字符串上下文键，则还必须使用 `StringLike` 条件运算符。如果有多个条件键值，则必须像数组一样用方括号括起来，例如 `"Key2":["Value2A", "Value2B"]`。

ForAllValues

`ForAllValues` 限定符测试请求上下文中每个成员的值是否与其后跟的条件运算符匹配。如果请求中的每个上下文键值均与策略中的一个上下文键值匹配，则条件返回 `true`。如果请求中没有上下文键，则它也会返回 `true`。

> **Important**
>
> 如果将 `ForAllValues` 与 `Allow` 效果一起使用，请小心谨慎，因为如果请求上下文中意外出现缺失的上下文键，则策略可能会过于宽松。您应始终在策略中包含具有 `false` 值的 `Null` 条件运算符，以检查上下文键是否存在且其值不为空。有关示例，请参阅根据标签键控制访问。

ForallValues 集合运算符示例

在以下示例中，ForAllValues 与 aws:TagKeys 一起使用，允许用户删除分配给 EC2 实例的特定标签。此策略仅允许用户删除 `environment` 和 `cost-center` 标签。您可以单独删除它们，也可以将其一起删除。请求中的标签键必须与策略中指定的键完全匹配。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteTags",
            "Resource": "arn:aws:ec2:us-east-1:111122223333:instance/*",
            "Condition": {
                "ForAllValues:StringEquals": {
                    "aws:TagKeys": [
                        "environment",
                        "cost-center"
                    ]
                },
                "Null": {
                    "aws:TagKeys": "false"
                }
            }
        }
```

```
    ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"ForAllValues:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }, "Null": { "aws:TagKeys": "false" }`	`aws:TagKeys: - environment`	匹配
`"ForAllValues:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }, "Null": { "aws:TagKeys": "false" }`	`aws:TagKeys: - cost-center`	匹配
`"ForAllValues:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }, "Null": {`	`aws:TagKeys: - environment - cost-center`	匹配

策略条件	请求上下文	结果
"aws:TagKeys": "false" }		
"ForAllValues:StringEquals": { "aws:TagKeys": ["environment", "cost-center"] }, "Null": { "aws:TagKeys": "false" }	aws:TagKeys: - environment - dept	不匹配
"ForAllValues:StringEquals": { "aws:TagKeys": ["environment", "cost-center"] }, "Null": { "aws:TagKeys": "false" }	请求上下文中没有 aws:TagKeys 。	不匹配

请注意，在最后一个示例中，结果为“无匹配”，因为当上下文键缺失时，空条件检查会阻止匹配。这是避免过于宽松的策略的最佳实践。

ForAnyValue

ForAnyValue 限定符测试请求上下文键值集中的至少一个成员是否与策略条件中上下文键值集中的至少一个成员匹配。如果请求中的任何一个上下文键值与策略中的任何一个上下文键值匹配，则条件返回 true。如果没有匹配的上下文键或者键不存在，则条件返回 false。

> **Important**
>
> 当使用具有 Deny 效果的 ForAnyValue 时，如果上下文键不存在于请求中，则策略的评估结果为不匹配。为了确保行为一致，请在策略中添加显式 Null 条件检查，以验证上下文键是否存在。有关更多信息，请参阅 用于检查条件键是否存在的条件运算符。

AnyValue 集合运算符示例

在以下示例中，ForAnyValue 与 aws:TagKeys 一起使用，允许用户删除分配给 EC2 实例的特定标签。如果请求中指定的标签键包括 environment 或 cost-center，则此策略允许用户删除实例的标签。请求可以包含策略中指定的标签键以外的其他标签键，但必须至少包含一个指定的键才能匹配条件。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "ec2:DeleteTags",
            "Resource": "arn:aws:ec2:us-east-1:111122223333:instance/*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:TagKeys": [
                        "environment",
                        "cost-center"
                    ]
                }
            }
        }
    ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"ForAnyValue:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }`	`aws:TagKeys: - environment`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }`	`aws:TagKeys: - cost-center`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }`	`aws:TagKeys: - environment - cost-center`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": [ "environment", "cost-center" ] }`	`aws:TagKeys: - environment - dept`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": [ "environment", "cost-center"`	`aws:TagKeys: - dept`	不匹配

策略条件	请求上下文	结果
] }		

"ForAnyValue:StringEquals": {
"aws:TagKeys": [
"environment",
"cost-center"
]
}
	请求上下文中没有 aws:TagKeys 。	不匹配

条件策略示例

在 IAM policy 中，您可以为单值和多值上下文键指定多个值，以便与请求上下文进行比较。以下一组策略示例演示了具有多个上下文键和值的策略条件。

> **Note**
>
> 如果您愿意提交策略供本参考指南采用，请使用该页面底部的 Feedback 按钮。有关 IAM 基于身份的策略的示例，请参阅 IAM 基于身份的策略示例。

条件策略示例：单值上下文键

- 具有单值上下文键的多个条件块。（查看此示例。）
- 具有多个单值上下文键和值的一个条件块。（查看此示例。）

条件策略示例：多值上下文键

- 使用条件集合运算符 ForAllValues 的拒绝策略。（查看此示例。）
- 使用条件集合运算符 ForAnyValue 的拒绝策略。（查看此示例。）

多值上下文键示例

以下一组策略示例演示了如何使用多值上下文键创建策略条件。

示例：使用条件集合运算符 ForAllValues 的拒绝策略

以下示例显示了当请求中包含特定标签键前缀时，如何使用基于身份的策略拒绝使用 IAM 标记操作。aws:TagKeys 的值包括用于部分字符串匹配的通配符 (*)。此策略包括具有上下文键 aws:TagKeys 的 ForAllValues 集合运算符，因为请求上下文键可以包含多个值。为了使上下文键 aws:TagKeys 匹配，请求上下文中的每个值必须与策略中的至少一个值匹配。

如果请求中没有上下文键，ForAllValues 集合运算符也会返回 true。

您可以在策略中包含一个值为 false 的 Null 条件运算符来检查请求中的上下文键是否存在且其值不为空，从而防止缺失的上下文键或具有空值的上下文键评估为 true。有关更多信息，请参阅 用于检查条件键是否存在的条件运算符。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

Example拒绝多值上下文键的单个策略条件值

在以下示例中，策略拒绝请求中 aws:TagKeys 的值不包含前缀 key1 的请求。请求上下文可以有多个值，但由于 ForAllValues 条件集合运算符，请求上下文中的所有标签键值都必须以前缀 key1 开头。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRestrictedTags",
      "Effect": "Deny",
      "Action": [
        "iam:Tag*",
        "iam:UnTag*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
```

```
        "ForAllValues:StringNotLike": {
          "aws:TagKeys": "key1*"
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。对于 Deny 语句，匹配表示被拒绝，不匹配表示不拒绝，因此它可能被另一个语句允许。

策略条件	请求上下文	结果
`"ForAllValues:StringNotLike": { "aws:TagKeys": "key1*" }`	`aws:TagKeys: - key1:legal`	不匹配 可能被另一个语句允许。
`"ForAllValues:StringNotLike": { "aws:TagKeys": "key1*" }`	`aws:TagKeys: - key1:hr - key1:personnel`	不匹配 可能被另一个语句允许。
`"ForAllValues:StringNotLike": { "aws:TagKeys": "key1*" }`	`aws:TagKeys: - key2:audit`	匹配
`"ForAllValues:StringNotLike": { "aws:TagKeys": "key1*" }`	请求上下文中没有 aws:TagKeys 。	匹配

Example拒绝多值上下文键的多个策略条件值

在以下示例中，策略拒绝请求中 aws:TagKeys 的值不包含前缀 key1 或 key2 的请求。请求上下文可以有多个值，但由于 ForAllValues 条件集合运算符，请求上下文中的所有标签键值都必须以前缀 key1 或 key2 开头。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyRestrictedTags",
      "Effect": "Deny",
      "Action": [
        "iam:Tag*",
        "iam:UnTag*"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "ForAllValues:StringNotLike": {
          "aws:TagKeys": [
            "key1*",
            "key2*"
          ]
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。对于 Deny 语句，匹配表示被拒绝，不匹配表示不拒绝，因此它可能被另一个语句允许。

策略条件	请求上下文	结果
"ForAllValues:Stri ngNotLike": {	aws:TagKeys:	不匹配

策略条件	请求上下文	结果
"aws:TagKeys": ["key1*", "key2*"] }	- key1:legal	可能被另一个语句允许。

"ForAllValues:StringNotLike": {
"aws:TagKeys": [
"key1*",
"key2*"
]
}
	aws:TagKeys:
	- key1:hr
	- key1:personnel
		不匹配
		可能被另一个语句允许。

"ForAllValues:StringNotLike": {
"aws:TagKeys": [
"key1*",
"key2*"
]
}
	aws:TagKeys:
	- key1:hr
	- key2:audit
		不匹配
		可能被另一个语句允许。

"ForAllValues:StringNotLike": {
"aws:TagKeys": [
"key1*",
"key2*"
]
}
	aws:TagKeys:
	- *key3*:legal
		匹配

"ForAllValues:StringNotLike": {
"aws:TagKeys": [
"key1*",
"key2*"
]
}
	请求上下文中没有 aws:TagKeys 。	匹配

示例：使用条件集合运算符 ForAnyValue 的拒绝策略

如果任何快照使用策略中指定的标签键之一（environment 或 webserver）进行标记，则以下基于身份的策略示例拒绝创建 EC2 实例卷的快照。此策略包括具有上下文键 aws:TagKeys 的 ForAnyValue 集合运算符，因为请求上下文键可以包含多个值。如果您的标记请求包含策略中指定的任何一个标签键值，则 aws:TagKeys 上下文键返回 true，以调用拒绝策略效果。

> **Important**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Deny",
      "Action": [
        "ec2:CreateSnapshot",
        "ec2:CreateSnapshots"
      ],
      "Resource": "arn:aws:ec2:us-west-2::snapshot/*",
      "Condition": {
        "ForAnyValue:StringEquals": {
          "aws:TagKeys": "webserver"
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。对于 Deny 语句，匹配表示被拒绝，不匹配表示不拒绝，因此它可能被另一个语句允许。

策略条件	请求上下文	结果
`"ForAnyValue:StringEquals": { "aws:TagKeys": "webserver" }`	`aws:TagKeys: - webserver`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": "webserver" }`	`aws:TagKeys: - environment - webserver - test`	匹配
`"ForAnyValue:StringEquals": { "aws:TagKeys": "webserver" }`	`aws:TagKeys: - environment - test`	不匹配 可能被另一个语句允许。
`"ForAnyValue:StringEquals": { "aws:TagKeys": "webserver" }`	请求上下文中没有 `aws:TagKeys` 。	不匹配 可能被另一个语句允许。

单值上下文键策略示例

以下一组策略示例演示了如何使用单值上下文键创建策略条件。

示例：具有单值上下文键的多个条件块

条件块有多个条件时，每个条件都有一个上下文键，所有上下文键必须解析为 true 才能调用所需的 `Allow` 或 `Deny` 效果。使用否定匹配条件运算符时，条件值的评估逻辑是相反的。

以下示例允许用户创建 EC2 卷并在创建卷期间将标签应用到卷。请求上下文必须包含上下文键 `aws:RequestTag/project` 的值，以及上下文键 `aws:ResourceTag/environment` 的值可以是除生产之外的任何内容。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "ec2:CreateVolume",
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:volume/*",
      "Condition": {
        "StringLike": {
          "aws:RequestTag/project": "*"
        }
      }
    },
    {
      "Effect": "Allow",
      "Action": "ec2:CreateTags",
      "Resource": "arn:aws:ec2:us-east-1:123456789012:*/*",
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceTag/environment": "production"
        }
      }
    }
  ]
}
```

请求上下文必须包含项目标签值，并且不能为生产资源创建以调用 `Allow` 效果。以下 EC2 卷已成功创建，因为项目名称为 `Feature3`，资源标签为 QA。

```
aws ec2 create-volume \
    --availability-zone us-east-1a \
    --volume-type gp2 \
    --size 80 \
```

```
    --tag-specifications 'ResourceType=volume,Tags=[{Key=project,Value=Feature3},
{Key=environment,Value=QA}]'
```

示例：具有多个单值上下文键和值的一个条件块

条件块包含多个上下文键并且每个上下文键具有多个值时，每个上下文键必须解析为 true，以便至少有一个键值能够调用所需的 Allow 或 Deny 效果。使用否定匹配条件运算符时，上下文键值的评估逻辑是相反的。

以下示例允许用户在 Amazon Elastic Container Service 集群上启动和运行任务。

- 对于 aws:RequestTag/environment 上下文键 AND，请求上下文必须包含 production 或 prod-backup。
- ecs:cluster 上下文键可确保在 default1 或 default2 ARN ECS 集群上运行任务。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "ecs:RunTask",
        "ecs:StartTask"
      ],
      "Resource": [
        "*"
      ],
      "Condition": {
        "StringEquals": {
          "aws:RequestTag/environment": [
            "production",
            "prod-backup"
          ]
        },
        "ArnEquals": {
          "ecs:cluster": [
            "arn:aws:ecs:us-east-1:111122223333:cluster/default1",
            "arn:aws:ecs:us-east-1:111122223333:cluster/default2"
          ]
```

```
        }
      }
    }
  ]
}
```

下表显示了 Amazon 如何根据请求中的条件键值来评估此策略。

策略条件	请求上下文	结果
`"StringEquals": { "aws:RequestTag/environment": [ "production", "prod-backup" ] }, "ArnEquals": { "ecs:cluster": [ "arn:aws:ecs:us-east-1:111122223333:cluster/default1", "arn:aws:ecs:us-east-1:111122223333:cluster/default2" ] }`	`aws:RequestTag: environment:production ecs:cluster: arn:aws:ecs:us-east-1:111122223333:cluster/default1`	匹配
`"StringEquals": { "aws:RequestTag/environment": [ "production", "prod-backup" ] }, "ArnEquals": { "ecs:cluster": [ "arn:aws:ecs:us-east-1:111122223333:cluster/default1",`	`aws:RequestTag: environment:prod-backup ecs:cluster: arn:aws:ecs:us-east-1:111122223333:cluster/default2`	匹配

策略条件	请求上下文	结果
"arn:aws:ecs:us-east-1:111122223333:cluster/default2"] }		

"StringEquals": {
"aws:RequestTag/environment": [
"production",
"prod-backup"
]
},
"ArnEquals": {
"ecs:cluster": [
"arn:aws:ecs:us-east-1:111122223333:cluster/default1",
"arn:aws:ecs:us-east-1:111122223333:cluster/default2"
]
}
	aws:RequestTag:
	webserver:production
	ecs:cluster:
	arn:aws:ecs:us-east-1:111122223333:cluster/default2
		不匹配

策略条件	请求上下文	结果
`"StringEquals": { "aws:RequestTag/environment": [ "production", "prod-backup" ] }, "ArnEquals": { "ecs:cluster": [ "arn:aws:ecs:us-east-1:111122223333:cluster/default1", "arn:aws:ecs:us-east-1:111122223333:cluster/default2" ] }`	请求上下文中没有 `aws:RequestTag` 。 `ecs:cluster arn:aws:ecs:us-east-1:111122223333:cluster/default2`	不匹配

IAM policy 元素：变量和标签

在编写策略时，如果您不知道资源或条件键的精确值，可以使用 Amazon Identity and Access Management (IAM) policy 变量作为占位符。

> **Note**
>
> 如果 Amazon 无法解析变量，这可能会导致整个语句无效。例如，如果您使用 `aws:TokenIssueTime` 变量，只有在请求者（IAM 角色）使用临时凭证进行身份验证时，该变量才会解析为一个值。要防止变量导致无效的语句，请使用 ...IfExists 条件运算符。

主题

- 简介
- 在策略中使用变量
- 作为策略变量的标签
- 您可以使用策略变量的位置
- 无值的策略变量

- 可以用于策略变量的请求信息
- 指定默认值
- 有关更多信息

简介

在 IAM policy 中，您可通过很多操作为要控制其访问权限的特定资源指定名称。例如，以下策略允许用户为 marketing 项目列出、读取 S3 存储桶 amzn-s3-demo-bucket 中的对象以及将对象写入该存储桶。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
      "Condition": {"StringLike": {"s3:prefix": ["marketing/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/marketing/*"]
    }
  ]
}
```

在某些情况下，您在编写策略时可能不知道资源的精确名称。您可能需要概括策略，这样无需为每个用户制作策略的唯一副本即可将该策略用于很多用户。我们建议您创建一个适用于该组中所有用户的单个组策略，而不是为每个用户创建一个单独的策略。

在策略中使用变量

您可以使用策略变量在策略中设置占位符，从而在策略中定义动态值。

变量使用 **$** 前缀标记，后面跟一对大括号 (**{ }**)，其中包含请求中值的变量名称。

评估策略时，策略变量将替换为来自请求中传递的条件上下文键的值。变量可用于基于身份的策略、资源策略、服务控制策略、会话策略和 VPC 端点策略。用作权限边界的基于身份的策略也支持策略变量。

全局条件上下文键可用作跨 Amazon 服务的请求中的变量。在与 Amazon 资源交互时，服务特定条件键也可以用作变量，但仅在针对支持它们的资源发出请求时才可用。有关每种 Amazon 服务和资源的可用上下文键列表，请参阅《服务授权参考》。某些情况下，您无法使用值填充全局条件上下文键。要了解有关每个键的更多信息，请参阅 Amazon 全局条件上下文键。

> **Important**
>
> - 密钥名称不区分大小写。例如，`aws:CurrentTime` 等同于 `AWS:currenttime`。
> - 您可以使用任何可用的单值条件密钥作为变量。您不能使用多值条件键作为变量。

以下示例显示了 IAM 角色或用户的策略，该策略用策略变量来替代特定资源名称。您可以利用 `aws:PrincipalTag` 条件键来重复使用此策略。完成策略评估后，仅当存储桶名称以 `team` 主体标签中的团队名称结尾时，`${aws:PrincipalTag/team}` 才允许操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": ["s3:ListBucket"],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${aws:PrincipalTag/team}/*"]}}
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/${aws:PrincipalTag/team}/*"]
```

```
    }
  ]
}
```

该变量使用 $ 前缀标记，后跟一对大括号 ({ })。在 ${ } 字符内，可以包含想要在策略中使用的请求中的值名称。本页稍后将讨论您可以使用的值。

有关此全局条件键的详细信息，请参阅全局条件键列表中的 aws:PrincipalTag/tag-key。

> **Note**
>
> 为了使用策略变量，您必须在语句中包含 Version 元素，而且版本必须设置为支持策略变量的版本。变量是在 2012-10-17 版本中引入的。较早版本的策略语言不支持策略变量。如果您未添加 Version 元素，且没有将它设为相应的版本日期，则系统会将变量 (如 ${aws:username}) 视为策略中的文字字符串。
> Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您更改 IAM 中的客户托管策略时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅the section called “Version”。要了解策略版本的更多信息，请参阅the section called “IAM 策略版本控制”。

允许主体从 S3 存储桶中的 /David 路径获取对象的策略如下所示：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:GetObject"
      ],
      "Resource": [
        "arn:aws:s3:::amzn-s3-demo-bucket/David/*"
      ]
    }
```

```
  ]
}
```

如果将此策略附加到用户 David，则该用户将获取自己 S3 存储桶中的对象，但您必须为每个用户创建包含该用户名称的单独策略，然后将每个策略附加到各个用户。

通过使用策略变量，您可以创建可重复使用的策略。以下策略允许用户在 aws:PrincipalTag 的标签键值与请求中传递的标签键 owner 值一致时，获取 Amazon S3 存储桶中的对象。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [{
    "Sid": "AllowUnlessOwnedBySomeoneElse",
    "Effect": "Allow",
    "Action": ["s3:GetObject"],
    "Resource": ["*"],
    "Condition": {
        "StringEquals": {
          "s3:ExistingObjectTag/owner": "${aws:PrincipalTag/owner}"
        }
      }
    }
  ]
}
```

如果您使用策略变量，而不是与此类似的用户名称，则无需为每个单独的用户创建单独的策略。在以下示例中，该策略被附加到一个将由产品经理使用临时安全凭证代入的 IAM 角色。当用户发出添加 Amazon S3 对象的请求时，IAM 将使用当前请求中的 dept 标签值替换 ${aws:PrincipalTag} 变量，然后评估策略。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
```

```
            "Sid": "AllowOnlyDeptS3Prefix",
            "Effect": "Allow",
            "Action": [
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket/${aws:PrincipalTag/dept}/*"
            ]
        }
    ]
}
```

作为策略变量的标签

在某些 Amazon 服务中，您可以将自己的自定义属性附加到这些服务创建的资源。例如，您可以将标签应用于 Amazon S3 存储桶或者 IAM 用户。这些标签是键值对。您要定义标签键名称以及与该键名称关联的值。例如，您可以创建一个具有 **department** 键和 **Human Resources** 值的标签。有关标记 IAM 实体的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。有关标记其他 Amazon 服务创建的资源的信息，请参阅该服务的文档。有关使用标签编辑器的信息，请参阅《Amazon Web Services 管理控制台 用户指南》中的使用标签编辑器。

您可以标记 IAM 资源来简化对您的 IAM 资源的发现、组织和跟踪。您也可以标记 IAM 身份以控制对资源或标记本身的访问。要了解有关使用标签控制访问的更多信息，请参阅 使用标签控制对 IAM 用户和角色的访问以及他们进行的访问。

您可以使用策略变量的位置

您可以在 Resource 元素中和 Condition 元素的字符串比较中使用策略变量。

资源元素

您可以在 Resource 元素中使用策略变量，但只能在 ARN 的资源部分中使用策略变量。ARN 的这一部分出现在第五个冒号 (:) 之后。不能使用变量来替换 ARN 中第五个冒号之前的部分，例如服务或账户。有关 ARN 格式的更多信息，请参见IAM ARN。

要将 ARN 的一部分替换为标签值，请用 ${ } 将前缀和键名称括起来。例如，以下 Resource 元素将仅指向名称等于请求用户的 department 标签中的值的存储桶。

```
"Resource": ["arn:aws::s3:::amzn-s3-demo-bucket/
${aws:PrincipalTag/department}"]
```

许多 Amazon 资源使用包含用户创建的名称的 ARN。以下 IAM policy 确保只有 access-project、access-application 和 access-environment 标签值匹配的目标用户才能修改其资源。此外，使用 * 通配符匹配，将可以允许自定义资源名称后缀。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessBasedOnArnMatching",
      "Effect": "Allow",
      "Action": [
        "sns:CreateTopic",
        "sns:DeleteTopic"],
      "Resource": ["arn:aws:sns:*:*:${aws:PrincipalTag/access-project}-
${aws:PrincipalTag/access-application}-${aws:PrincipalTag/access-environment}-*"
      ]
    }
  ]
}
```

条件元素

您可以在任何涉及字符串运算符或 ARN 运算符的条件下对 Condition 值使用策略变量。字符串运算符包括 StringEquals、StringLike 和 StringNotLike。ARN 运算符包括 ArnEquals 和 ArnLike。不能将策略变量与其他运算符（如Numeric、Date、Boolean、Binary、IP Address 或 Null 运算符）一起使用。有关条件运算符的更多信息，请参阅IAM JSON 策略元素：条件运算符。

当引用 Condition 元素表达式中的标签时，请将相关的前缀和标签键用作条件键。然后，使用要在条件值中测试的值。

例如，以下示例策略仅在将标签 costCenter 附加到用户时才允许用户拥有完全访问权限。该标签还必须具有值 12345 或 67890。如果该标签没有值或具有任何其他值，则请求会失败。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
```

```
    {
      "Effect": "Allow",
      "Action": [
          "iam:*user*"
       ],
      "Resource": "*",
      "Condition": {
        "StringLike": {
          "iam:ResourceTag/costCenter": [ "12345", "67890" ]
        }
      }
    }
  ]
}
```

无值的策略变量

当策略变量引用的条件上下文键没有值或不存在于请求的授权上下文中时，该值实际上为空。没有相等或相似的值。在以下情况下，条件上下文键可能不存在于授权上下文中：

- 您在对不支持该条件键的资源的请求中使用服务特定条件上下文键。
- IAM 主体、会话、资源或请求上的标签不存在。
- 为 Amazon 全局条件上下文密钥 中的每个全局条件上下文键列出的其他情况。

当您在 IAM policy 的条件元素中使用没有值的变量时，IAM JSON 策略元素：条件运算符（如 `StringEquals` 或 `StringLike`）不匹配，并且策略声明不生效。

反向条件运算符（如 `StringNotEquals` 或 `StringNotLike`）确实与空值相匹配，因为它们测试的条件键的值不等于或不类似于实际空值。

在下例中，`aws:principaltag/Team` 必须等于 `s3:ExistingObjectTag/Team` 才能允许访问。未设置 `aws:principaltag/Team` 时会被显式拒绝访问。如果将授权上下文中没有值的变量用作策略的 `Resource` 或 `NotResource` 元素的一部分，则包含无值的策略变量的资源与任何资源都不匹配。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
   {
    "Effect": "Deny",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
      "StringNotEquals": {
        "s3:ExistingObjectTag/Team": "${aws:PrincipalTag/Team}"
       }
      }
    }
  ]
}
```

可以用于策略变量的请求信息

您可以使用 JSON 策略的 Condition 元素将请求上下文中的键与您在策略中指定的键值进行比较。当您使用策略变量时，Amazon 使用请求上下文密钥中的值替换策略中的变量。

主体键值

aws:username、aws:userid 和 aws:PrincipalType 的值取决于发起请求的主体的类型。例如，可能使用 IAM 用户、IAM 角色或 Amazon Web Services 账户根用户 的凭证发出请求。以下列表为不同类型的主体显示这些键的值。

- Amazon Web Services 账户根用户
  - aws:username: (不存在)
  - aws:userid：Amazon Web Services 账户 ID
  - aws:PrincipalType: Account
- IAM 用户
  - aws:username: *IAM ####*
  - aws:userid: 唯一 ID
  - aws:PrincipalType: User
- Amazon STS 联合用户主体
  - aws:username: (不存在)
  - aws:userid: *##*:*########*
  - aws:PrincipalType: FederatedUser

- OIDC 和 SAML 联合主体

> **Note**
>
> 有关在使用 OIDC 联合身份验证时可用的策略键的信息，请参阅 OIDC 联合身份验证。

  - `aws:username`: (不存在)
  - `aws:userid`: (不存在)
  - `aws:PrincipalType`: `AssumedRole`
- 担任的角色
  - `aws:username`: (不存在)
  - `aws:userid`: *`## ID`*`:##########`
  - `aws:PrincipalType`: `Assumed role`
- 分配给 Amazon EC2 实例的角色
  - `aws:username`: (不存在)
  - `aws:userid`: *`## ID`*`:EC2 ## ID`
  - `aws:PrincipalType`: `Assumed role`
- 匿名调用者（仅限 Amazon SQS、Amazon SNS 和 Amazon S3）
  - `aws:username`: (不存在)
  - `aws:userid`: (不存在)
  - `aws:PrincipalType`: `Anonymous`

对于此列表，请注意以下事项：

- 不存在 表示值在当前请求信息中不存在，并且匹配该值的任何尝试都失败并导致语句无效。
- *`## ID`* 是在创建时分配给每个角色的唯一标识符。可以使用 Amazon CLI 命令显示角色 ID：`aws iam get-role --role-name` *`rolename`*
- *`########`*和*`##########`* 是进行调用以获取临时凭证时调用进程 (例如应用程序或服务) 传递的名称。
- *`ec2-instance-id`* 是在实例启动时分配给它的值，显示在 Amazon EC2 控制台的 Instances（实例）页面上。您还可以通过运行 Amazon CLI 命令来显示实例 ID：`aws ec2 describe-instances`

联合主体的请求中的可用信息

联合主体是使用系统而不是 IAM 验证的用户。例如，公司可能拥有调用 Amazon 的供内部使用的应用程序。为使用该应用程序的每位公司用户都提供 IAM 身份可能不现实。相反，该公司可能使用具有单一 IAM 身份的代理 (中间层) 应用程序，或该公司可能使用 SAML 身份提供程序 (IdP)。代理应用程序或 SAML IdP 会使用企业网络对单个用户进行身份验证。然后，代理应用程序可使用其 IAM 身份获取单个用户的临时安全凭证。实际上，SAML IdP 可以将身份信息交换为 Amazon 临时安全凭证。然后，临时凭证即可用于访问 Amazon 资源。

类似地，您可以创建用于移动设备的应用程序，该应用程序需要访问 Amazon 资源。在这种情况下，您可以使用 OIDC 联合身份验证，其中应用程序使用知名身份提供者（如 Login with Amazon、Amazon Cognito、Facebook 或 Google）对用户进行身份验证。随后，应用程序可以使用这些提供商提供的用户身份验证信息来获取访问 Amazon 资源的临时安全凭证。

使用 OIDC 联合身份验证的推荐方法是利用 Amazon Cognito 和 Amazon 移动 SDK。有关更多信息，请参阅下列内容：

- Amazon Cognito 用户指南
- 临时凭证的常见情形

特殊字符

有几个特殊预定义策略变量具有固定值，可用于表示字符 (这些字符本身有特殊的含义)。如果这些特殊字符是您尝试匹配的字符串的一部分，而您原样插入这些字符，则不能正确进行解释。例如，在字符串中插入 * 星号会解释为与任何字符匹配的通配符 (而不是解释为文本 *)。这种情况下，可以使用以下预定义策略变量：

- ${*} - 在需要 *（星号）字符的位置使用。
- ${?} - 在需要 ?（问号）字符的位置使用。
- ${$} - 在需要 $（美元符号）字符的位置使用。

在任何可以使用常规策略变量的字符串中，都可以使用这些预定义策略变量。

指定默认值

要向变量添加默认值，请用单引号 (' ') 括起默认值，并用逗号和空格 (,) 分隔变量文本和默认值。

例如，如果使用主体标记 `team=yellow`，他们可以访问名为 `amzn-s3-demo-bucket-yellow` 的 `ExampleCorp's` Amazon S3 存储桶。使用此资源的策略可能允许团队成员访问自己的团队存储桶，

但不能访问其他团队的存储桶。对于没有团队标签的用户，它将原定设置值设为 company-wide 存储桶名称。这些用户只能访问 amzn-s3-demo-bucket-company-wide 存储桶，他们可以在其中查看广泛的信息，例如加入团队的说明。

```
"Resource":"arn:aws:s3:::amzn-s3-demo-bucket-${aws:PrincipalTag/team, 'company-wide'}"
```

有关更多信息

欲了解更多有关策略的信息，请参阅：

- Amazon Identity and Access Management 中的策略和权限
- IAM 基于身份的策略示例
- IAM JSON 策略元素参考
- 策略评估逻辑
- OIDC 联合身份验证

IAM JSON 策略元素：支持的数据类型

本节列出了在 JSON 策略中指定值时支持的数据类型。策略语言并不针对每个策略元素支持所有类型；有关各个元素的信息，请参阅前面各部分。

- 字符串
- 数字 (整数和浮动值)
- 布尔值
- Null
- Lists
- 地图
- 结构 (仅为嵌套映射)

以下图表将各个数据类型映射至串行化。注意所有策略必须为 UTF-8。有关 JSON 数据类型的信息，请转到 RFC 4627。

Type	JSON
字符串	字符串

Type	JSON
整数	数字
浮点型	数字
布尔值	真假
Null	null
日期	字符串，符合 W3C Profile of ISO 8601
IpAddress	字符串，符合 RFC 4632
列表	数组
对象	对象

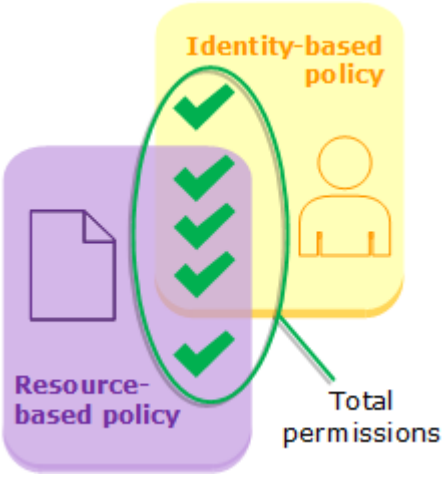
策略评估逻辑

在主体尝试使用 Amazon Web Services 管理控制台、Amazon API 或 Amazon CLI 时，该主体将向 Amazon 发送请求。在 Amazon 服务收到请求时，Amazon 会完成几个步骤来确定是允许还是拒绝该请求。

1. 身份验证 - Amazon 首先对发出请求的主体进行身份验证（如有必要）。有一些服务（如 Amazon S3）不需要此步骤，它们允许来自匿名用户的某些请求。
2. 处理请求上下文 - Amazon 处理在请求中收集的信息以确定应用于请求的策略。
3. Amazon 执行代码逻辑如何评估允许或拒绝访问的请求：Amazon 会评估所有策略类型，策略的顺序会影响策略的评估方式。随后，Amazon 根据请求上下文处理策略，决定是允许还是拒绝请求。

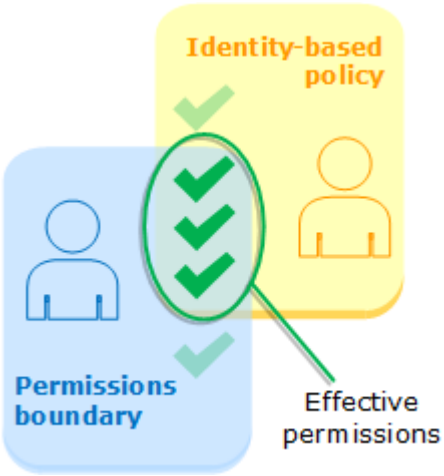
评估基于身份的策略以及基于资源的策略

基于身份的策略和基于资源的策略向策略所附加到的身份或资源授予权限。在 IAM 实体（用户或角色）请求访问同一账户中的资源时，Amazon 评估基于身份的策略和基于资源的策略授予的所有权限。生成的权限是指两种类型的权限的联合。如果基于身份的策略和/或基于资源的策略允许此操作，则 Amazon 允许执行该操作。其中任一项策略中的显式拒绝将覆盖允许。



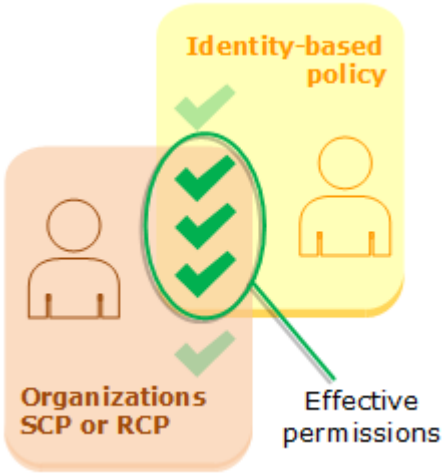
评估具有权限边界的基于身份的策略

在 Amazon 评估用户的基于身份的策略和权限边界时，生成的权限是这两种类别的交集。这意味着，当您通过现有基于身份的策略向用户添加权限边界时，您可能会减少用户可以执行的操作。或者，当您从用户删除权限边界时，您可能会增加用户可以执行的操作。其中任一项策略中的显式拒绝将覆盖允许。要查看有关如何使用权限边界评估其他策略类型的信息，请参阅评估具有边界的有效权限。



评估具有 Amazon Organizations SCP 或 RCP 的基于身份的策略

当一个用户属于组织成员账户且能够访问未配置基于资源的策略的资源，生成的权限是该用户的策略、服务控制策略（SCP）和资源控制策略（RCP）的交集。这意味着所有三种策略类型均必须允许某项操作。基于身份的策略、SCP 或 RCP 中的显式拒绝会覆盖该允许。



您可以在 Amazon Organizations 中了解您的账户是否为某个组织的成员。组织成员可能会受 SCP 或 RCP 的影响。要使用 Amazon CLI 命令或 Amazon API 操作查看该数据，您必须具有 Amazon Organizations 实体的 `organizations:DescribeOrganization` 操作的权限。您必须具有额外的权限才能在 Amazon Organizations 控制台中执行该操作。要了解 SCP 或 RCP 是否拒绝访问特定的请求或更改您的有效权限，请与您的 Amazon Organizations 管理员联系。

处理请求上下文

在 Amazon 评估和授权请求时，它会将请求信息汇编到请求上下文中。请求上下文包含可用于策略评估的任何信息。

- 主体：发送请求的用户、角色或联合用户主体。有关主体的信息包括与该主体关联的策略。
- 操作 – 主体希望执行的一个或多个操作。
- 资源 – 对其执行操作的一个或多个 Amazon 资源对象。
- 资源数据 – 与请求的资源相关的数据。这可能包括 DynamoDB 表名称或 Amazon EC2 实例上的标签等信息。
- 环境数据 – 有关 IP 地址、用户代理、SSL 启用状态或当天时间的信息。

将此信息与适用的策略进行比较，以确定是允许还是拒绝该请求。您可以使用主体、操作、资源和条件 (PARC) 模型组织此属性信息，以更好地了解如何评估 Amazon 策略。

了解 PARC 模型

PARC 模型基于策略语言中的四个 JSON 元素来表示请求上下文：

- Principal – 发出请求的实体。主体代表人类用户或编程工作负载，可对身份进行验证，然后向其授予在 Amazon Web Services 账户 中执行操作的权限。

- Action – 正在执行的操作。该操作通常会映射到 API 操作。
- Resource – 正在对其执行操作的 Amazon 资源。
- Condition – 允许请求必须满足的附加约束。

下面展示了 PARC 模型如何表示请求上下文的示例：

```
Principal: AIDA123456789EXAMPLE
Action: s3:CreateBucket
Resource: arn:aws:s3:::amzn-s3-demo-bucket1
Context:
- aws:UserId=AIDA123456789EXAMPLE:BobsSession
- aws:PrincipalAccount=123456789012
- aws:PrincipalOrgId=o-example
- aws:PrincipalARN=arn:aws:iam::AIDA123456789EXAMPLE:role/HR
- aws:MultiFactorAuthPresent=true
- aws:CurrentTime=...
- aws:EpochTime=...
- aws:SourceIp=...
- aws:PrincipalTag/dept=123
- aws:PrincipalTag/project=blue
- aws:RequestTag/dept=123
```

请求上下文的重要性

理解请求上下文及其与策略评估的交互方式对于以下方面至关重要：

- 排查访问问题
- 设计有效且安全的策略
- 了解策略授予的权限的全部范围
- 预测不同情景下的策略评估结果

通过使用 PARC 模型可视化请求上下文，您可以更轻松地了解 Amazon 如何做出授权决策并更有效地设计策略。

Amazon 如何使用请求上下文

评估策略时，Amazon 会将请求上下文中的信息与所有适用策略中指定的信息进行比较。这包括基于身份的策略、基于资源的策略、IAM 权限边界、组织 SCP、组织 RCP 和会话策略。

对于每种策略类型，Amazon 使用请求上下文来检查以下各项：

- 根据主体，策略是否适用于请求。
- 是否允许对指定的资源执行请求的操作。
- 请求上下文是否满足策略中指定的任何条件。

Amazon 如何评估策略取决于适用于请求上下文的策略类型。可在单个 Amazon Web Services 账户中使用这些策略类型。有关这些策略类型的更多信息，请参阅Amazon Identity and Access Management 中的策略和权限。要了解 Amazon 如何评估跨账户访问策略，请参阅跨账户策略评估逻辑。

- Amazon Organizations 资源控制策略（RCP）：Amazon Organizations RCP 指定组织或组织单位（OU）中账户内的资源的最大权限。RCP 适用于成员账户中的资源，并影响主体的有效权限，包括 Amazon Web Services 账户根用户，无论主体是否属于您的组织。RCP 不适用于组织管理账户中的资源和服务相关角色进行的调用。如果存在 RCP，则基于身份和基于资源的策略授予您的成员账户中的资源的权限仅在 RCP 允许操作时才有效。
- Amazon Organizations 服务控制策略（SCP）：Amazon Organizations SCP 指定组织或组织单位（OU）中账户内的主体的最大权限。SCP 适用于成员账户中的主体，包括每个 Amazon Web Services 账户根用户。如果 SCP 存在，仅当 SCP 允许该操作时，基于身份和基于资源的策略向成员账户中的主体授予的权限才有效。唯一的例外是组织管理账户中的主体和服务相关角色。
- 基于资源的策略：基于资源的策略为策略中指定的主体授予权限。权限定义主体可以对策略附加到的资源执行哪些操作。
- IAM 权限边界 – 权限边界是一项功能，它设置基于身份的策略可以授予 IAM 实体（用户或角色）的最大权限。当您设置实体的权限边界时，该实体只能执行其基于身份的策略和其权限边界同时允许的操作。某些情况下，权限边界中的隐式拒绝可能会限制由基于资源的策略所授予的权限。有关更多信息，请参阅 Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。
- 基于身份的策略 — 基于身份的策略附加到 IAM 身份（用户、用户组或角色）并向 IAM 实体（用户和角色）授予权限。如果仅基于身份的策略适用于请求，则 Amazon 检查所有这些策略以找到至少一个 `Allow` 。
- 会话策略：会话策略是一些策略，在以编程方式为角色或联合用户会话创建临时会话时，这些策略将作为参数进行传递。要以编程方式创建角色会话，请使用 `AssumeRole*` API 操作之一。在执行该操作并传递会话策略时，生成的会话的权限是 IAM 实体的基于身份的策略与会话策略的交集。要创建联合用户会话，您需要使用 IAM 用户的访问密钥以编程方式调用 `GetFederationToken` API 操作。有关更多信息，请参阅 会话策略。

请记住，任一项策略中的显式拒绝将覆盖允许。

> **Note**
>
> 借助 Amazon Organizations 声明式策略，您可在整个组织中针对给定 Amazon Web Services 服务 大规模集中声明和强制执行所需的配置。声明式策略直接应用于服务级别，因此这些策略不会直接影响策略评估请求，也不会包含在请求上下文中。有关更多信息，请参阅《Amazon Organizations User Guide》中的 Declarative policies。

使用 PARC 模型的策略评估示例

为了说明请求上下文如何与策略评估进行交互，请考虑示例策略：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": "s3:CreateBucket",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1",
      "Condition": {
        "StringEquals": {
          "aws:PrincipalTag/dept": "123"
        }
      }
    }
  ]
}
```

在此示例中，仅当请求上下文包含 `aws:PrincipalTag/dept` 值“123”且资源与 `amzn-s3-demo-bucket1` 存储桶名称匹配时，策略才会允许 `CreateBucket` 操作。下表显示了 Amazon 如何使用请求上下文来评估此策略并做出授权决策。

Policy	请求上下文	评估结果
`"StringEquals": { "aws:PrincipalTag/dept": "123"`	`Principal: AIDA12345 6789EXAMPLE`	匹配

Policy	请求上下文	评估结果
} 	Action: s3:Create Bucket Resource: arn:aws:s 3:::amzn-s3-demo-b ucket1 Context: - aws:PrincipalTag/ dept=123	

"StringEquals": {
"aws:PrincipalTag/
dept": "123"
}
	Principal: AIDA12345
	6789EXAMPLE
	Action: s3: *DeleteBuc*
	ket
	Resource: arn:aws:s
	3:::amzn-s3-demo-b
	ucket1
	Context:
	- aws:PrincipalTag/
	dept=123
		不匹配

"StringEquals": {
"aws:PrincipalTag/
dept": "123"
}
	Principal: AIDA12345
	6789EXAMPLE
	Action: s3:Create
	Bucket
	Resource: arn:aws:s
	3:::amzn-s3-demo-b
	ucket1
	Context:
	- aws:PrincipalTag/
	dept= *321*
		不匹配

Policy	请求上下文	评估结果

`"StringEquals": { "aws:PrincipalTag/dept": "123" }`	`Principal: AIDA123456789EXAMPLE`
	`Action: s3:CreateBucket`
	`Resource: arn:aws:s3:::amzn-s3-demo-bucket1`
	`Context:`
	请求中没有 `aws:PrincipalTag` 。
		不匹配

Amazon 执行代码逻辑如何评估允许或拒绝访问的请求

Amazon 执行代码决定是应该允许还是拒绝发送到 Amazon 的请求。Amazon 评估适用于请求上下文的所有策略。下面概述了 Amazon 策略评估逻辑。

- 默认情况下，除 Amazon Web Services 账户根用户 外，所有请求都被隐式否定，具有完全访问权限。
- 请求必须由遵循以下被允许的评估逻辑的策略或一组策略明确允许。
- 显式拒绝将会覆盖显式允许。

策略评估可能会有所不同，具体视请求是在单个账户内还是跨账户请求中而定。有关如何针对单一账户中的 IAM 角色或用户作出策略评估决策的详细信息，请参阅对单个账户内的请求进行策略评估。有关如何针对跨账户请求作出策略评估决策的详细信息，请参阅跨账户策略评估逻辑。

- Deny evaluation（拒绝评估）- 默认情况下，所有请求都被拒绝。这称为隐式拒绝。Amazon 执行代码评估账户中应用于请求的所有策略。其中包含 Amazon Organizations SCP 和 RCP、基于资源的策略、基于身份的策略、IAM 权限边界和会话策略。在所有这些策略中，执行代码查找应用于请求的 `Deny` 语句。这称为显式拒绝。如果执行代码找到一个适用的显式拒绝，则该执行代码将返回 Deny 的最终决定。如果没有显式拒绝，则执行代码评估会继续。
- Amazon Organizations RCP：执行代码会评估适用于请求的 Amazon Organizations 资源控制策略（RCP）。RCP 适用于附加了 RCP 的账户的资源。如果执行代码没有在 RCP 中找到任何适用的 `Allow` 语句，则执行代码将返回 Deny 的最终决定。请注意，当启用 RCP 时，系统会自动创建一个

名为 RCPFullAWSAccess 的 Amazon 托管式策略，并将其附加到组织中的每个实体，包括根、每个 OU 和 Amazon Web Services 账户。RCPFullAWSAccess 无法分离，因此始终将有一个 Allow 语句。如果没有 RCP，或者 RCP 允许所请求的操作，则执行代码评估继续适用。

- Amazon Organizations SCP：执行代码会评估适用于请求的 Amazon Organizations 服务控制策略（SCP）。SCP 适用于附加 SCP 的账户的主体。如果执行代码没有在 SCP 中找到任何适用的 Allow 语句，则执行代码将返回 Deny 的最终决定。如果没有 SCP，或者 SCP 允许所请求的操作，则执行代码评估继续。
- 基于资源的策略 — 在同一账户中，基于资源的策略会对策略评估产生不同的影响，具体取决于访问资源的主体类型以及基于资源的策略中允许的主体。根据主体的类型，Allow 在基于资源的策略中可能会导致最终决定 Allow，即使基于身份的策略、权限边界或会话策略中存在隐式拒绝。

  对于大多数资源，您只需要在基于身份的策略或基于资源的策略中明确 Allow 主体访问权限。IAM 角色信任策略和 KMS 密钥策略是此逻辑的例外，因为它们必须明确允许主体的访问权限。非 IAM 和 Amazon KMS 的基于资源的服务策略也可能要求在同一账户中作出显式 Allow 声明，才能授予访问权限。有关详细信息，请参阅您使用的特定服务的文档。

  对于单个账户策略评估请求，当指定的主体是 IAM 用户、IAM 角色或会话主体时，基于资源的策略逻辑会与其他策略类型有所不同。会话主体包括 IAM 角色会话或 Amazon STS 联合用户主体。如果基于资源的策略直接向提出请求的 IAM 用户或会话主体授予权限，则基于身份的策略、权限边界或会话策略中的隐式拒绝不会影响最终决策。

  - IAM 角色 — 授予 IAM 角色 ARN 权限的基于资源的策略受权限边界或会话策略中隐式拒绝的限制。可以在 Principal 元素或 aws:PrincipalArn 条件键中指定角色 ARN。在这两种情况下，提出请求的主体都是 IAM 角色会话。

    权限边界和会话策略不限制在 Principal 元素中使用带通配符 (*) 的 aws:PrincipalArn 条件键授予权限，除非基于身份的策略包含显式拒绝。有关更多信息，请参阅 IAM 角色主体。

    角色示例 ARN

    ```
    arn:aws:iam::111122223333:role/examplerole
    ```

  - IAM 角色会话— 在同一账户中，向 IAM 角色会话 ARN 授予权限的基于资源的策略直接向担任的角色会话授予权限。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。当您担任角色并提出请求时，发出请求的主体是 IAM 角色会话 ARN，而不是角色本身的 ARN。有关更多信息，请参阅 角色会话主体。

    示例角色会话 ARN

```
arn:aws:sts::111122223333:assumed-role/examplerole/examplerolesessionname
```

- IAM 用户 – 在同一账户中，向 IAM 用户 ARN（不是联合身份用户会话）授予权限的基于资源的策略不受基于身份的策略或权限边界中隐式拒绝的限制。

  示例 IAM 用户 ARN

  ```
  arn:aws:iam::111122223333:user/exampleuser
  ```

- Amazon STS 联合用户主体 – 联合用户会话是通过调用 GetFederationToken 创建的会话。当联合身份用户发出请求时，发出请求的主体是联合身份用户 ARN，而不是联合身份的 IAM 用户的 ARN。在同一个账户中，将权限授予联合身份用户 ARN 的基于资源的策略直接将权限授予会话。直接授予会话的权限不受基于身份的策略、权限边界或会话策略中的隐式拒绝的限制。

  但是，如果基于资源的策略向联合身份的 IAM 用户的 ARN 授予权限，则联合身份用户在会话期间发出的请求将受权限边界或会话策略中隐式拒绝的限制。

  示例联合用户会话 ARN

  ```
  arn:aws:sts::111122223333:federated-user/exampleuser
  ```

- 基于身份的策略：执行代码会检查主体的基于身份的策略。对于 IAM 用户，这包括用户策略和来自用户所属组的策略。如果没有基于身份的策略或者基于身份的策略中没有允许请求动作的语句，那么请求被隐式否定，执行代码返回 Deny 的最终决定。如果任何适用的基于身份的策略中的任何语句都允许请求的动作，则代码评估继续适用。
- IAM 权限边界：执行代码会检查主体使用的 IAM 实体是否具有权限边界。如果用于设置权限边界的策略不允许所请求的操作，则请求会被隐式拒绝。执行代码将返回拒绝最终决定。如果没有权限边界，或者权限边界允许所请求的操作，则代码评估继续适用。
- 会话策略：执行代码检查主体是否为会话主体。会话主体包括 IAM 角色会话或者 Amazon STS 联合用户会话。如果主体不是会话主体，执行代码将返回 Allow 的最终决定。

  对于会话主体，执行代码检查请求中是否传递了会话策略。您可以传递会话策略，同时使用 Amazon CLI 或 Amazon API 为某个角色或 Amazon STS 联合用户主体获取临时凭证。如果您没有传递会话策略，则会创建默认会话策略，执行代码返回 Allow 的最终决定。

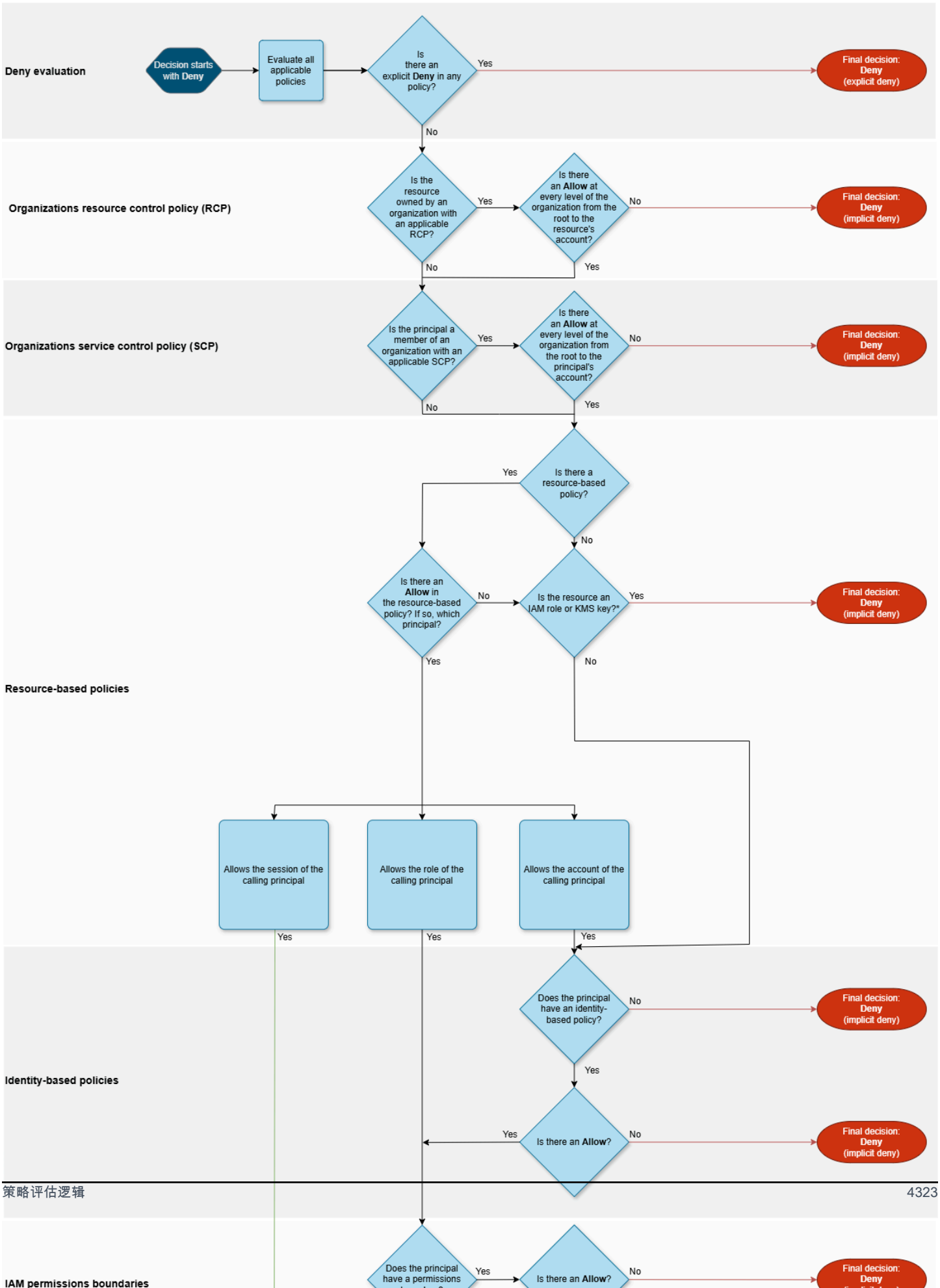
  - 如果会话策略存在但不允许所请求的操作，则请求会被隐式拒绝。执行代码将返回拒绝最终决定。
  - 执行代码将检查主体是否为角色会话。如果主体是角色会话，那么请求是已允许。否则，请求被隐式拒绝，执行代码返回 Deny 的最终决定。

- 如果会话策略在场并允许请求的动作，那么执行代码返回一个 Allow 的最终决策。

对单个账户内的请求进行策略评估

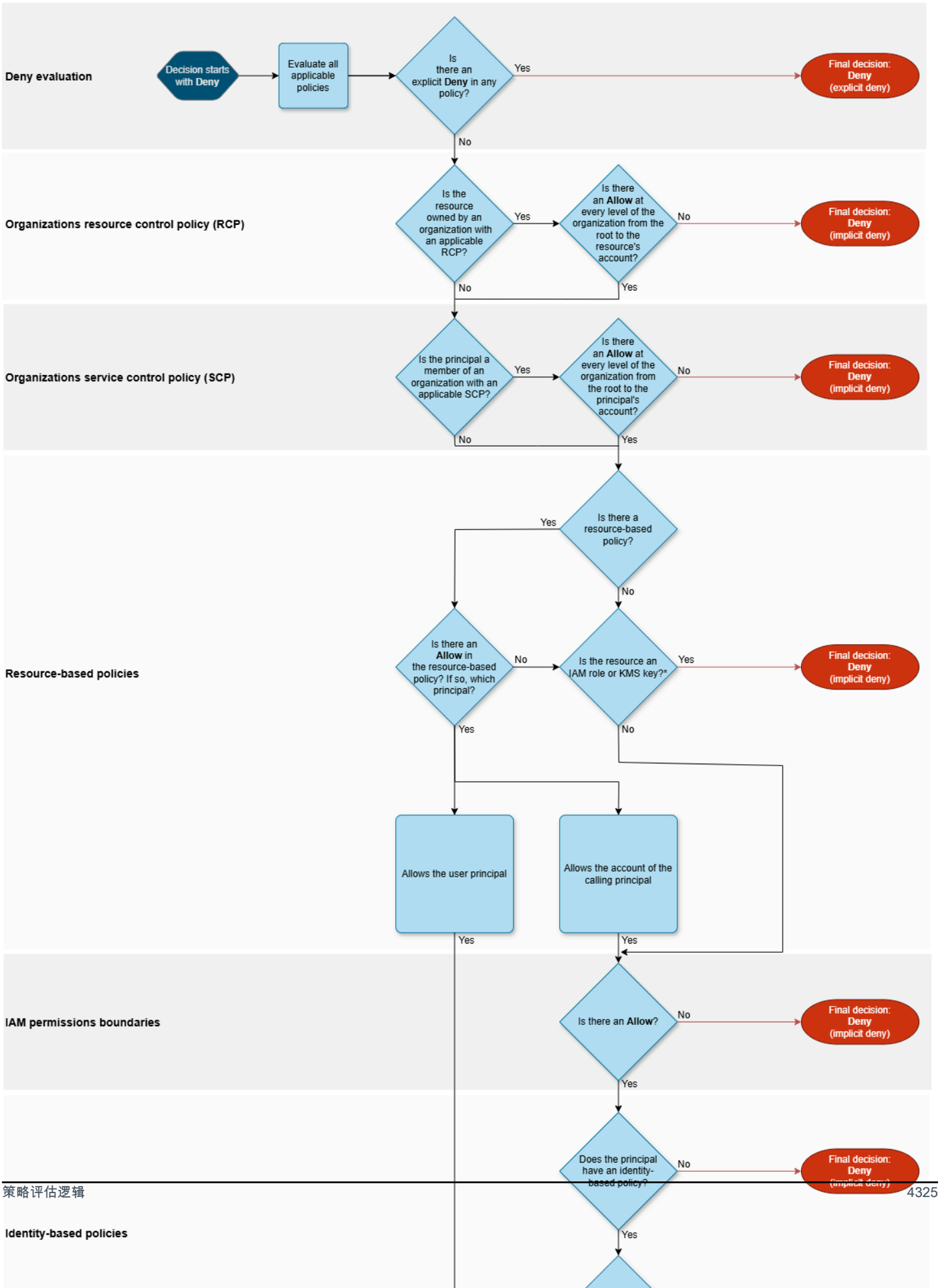
对 IAM 角色进行策略评估

以下流程图详细介绍了如何针对单一账户中的 IAM 角色作出策略评估决策。


Deny evaluation
Decision starts with Deny
Evaluate all applicable policies
Is there an explicit Deny in any policy?
Yes
Final decision: Deny (explicit deny)
No
Organizations resource control policy (RCP)
Is the resource owned by an organization with an applicable RCP?
Yes
Is there an Allow at every level of the organization from the root to the resource's account?
No
Final decision: Deny (implicit deny)
No
Yes
Organizations service control policy (SCP)
Is the principal a member of an organization with an applicable SCP?
Yes
Is there an Allow at every level of the organization from the root to the principal's account?
No
Final decision: Deny (implicit deny)
No
Yes
Resource-based policies
Yes
Is there a resource-based policy?
No
Is there an Allow in the resource-based policy? If so, which principal?
No
Is the resource an IAM role or KMS key?*
Yes
Final decision: Deny (implicit deny)
Yes
No
Allows the session of the calling principal
Allows the role of the calling principal
Allows the account of the calling principal
Yes
Yes
Yes
Identity-based policies
Does the principal have an identity-based policy?
No
Final decision: Deny (implicit deny)
Yes
Yes
Is there an Allow?
No
Final decision: Deny (implicit deny)
IAM permissions boundaries
Does the principal have a permissions
Yes
Is there an Allow?
No
Final decision: Deny

对 IAM 用户进行策略评估

以下流程图详细介绍了如何针对单一账户中的 IAM 用户作出策略评估决策。


Deny evaluation
Decision starts with Deny
Evaluate all applicable policies
Is there an explicit Deny in any policy?
Yes
Final decision: Deny (explicit deny)
No
Organizations resource control policy (RCP)
Is the resource owned by an organization with an applicable RCP?
Yes
Is there an Allow at every level of the organization from the root to the resource's account?
No
Final decision: Deny (implicit deny)
No
Yes
Organizations service control policy (SCP)
Is the principal a member of an organization with an applicable SCP?
Yes
Is there an Allow at every level of the organization from the root to the principal's account?
No
Final decision: Deny (implicit deny)
No
Yes
Resource-based policies
Is there a resource-based policy?
Yes
No
Is there an Allow in the resource-based policy? If so, which principal?
No
Is the resource an IAM role or KMS key?*
Yes
Final decision: Deny (implicit deny)
Yes
No
Allows the user principal
Allows the account of the calling principal
Yes
IAM permissions boundaries
Is there an Allow?
No
Final decision: Deny (implicit deny)
Yes
Identity-based policies
Does the principal have an identity-based policy?
No
Final decision: Deny (implicit deny)
Yes

基于身份的策略和基于资源的策略评估示例

策略的最常见类型是基于身份的策略和基于资源的策略。当请求访问资源时，Amazon 会评估策略为同一账户中至少一个允许所授予的所有权限。任一项策略中的显式拒绝将覆盖允许。

> **Important**
>
> 如果同一账户中基于身份的策略或基于资源的策略其中一个允许该请求，而另一个不允许，仍可允许该请求。

假定 Carlos 具有用户名 `carlossalazar`，他尝试将文件保存到 `amzn-s3-demo-bucket-carlossalazar-logs` Amazon S3 存储桶。

还假定将以下策略附加到 `carlossalazar` IAM 用户。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": [
                "s3:GetBucketLocation",
                "s3:GetAccountPublicAccessBlock",
                "s3:ListAccessPoints",
                "s3:ListAllMyBuckets"
            ],
            "Resource": "arn:aws:s3:::*"
        },
        {
            "Sid": "AllowS3Self",
            "Effect": "Allow",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-carlossalazar/*",
                "arn:aws:s3:::amzn-s3-demo-bucket-carlossalazar"
            ]
        },
```

```
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": "arn:aws:s3:::*log*"
        }
    ]
}
```

此策略中的 AllowS3ListRead 语句允许 Carlos 查看账户中的所有存储桶的列表。AllowS3Self 语句允许 Carlos 完全访问与其用户名同名的存储桶。DenyS3Logs 语句拒绝 Carlos 访问名称中包含 log 的任何 S3 存储桶。

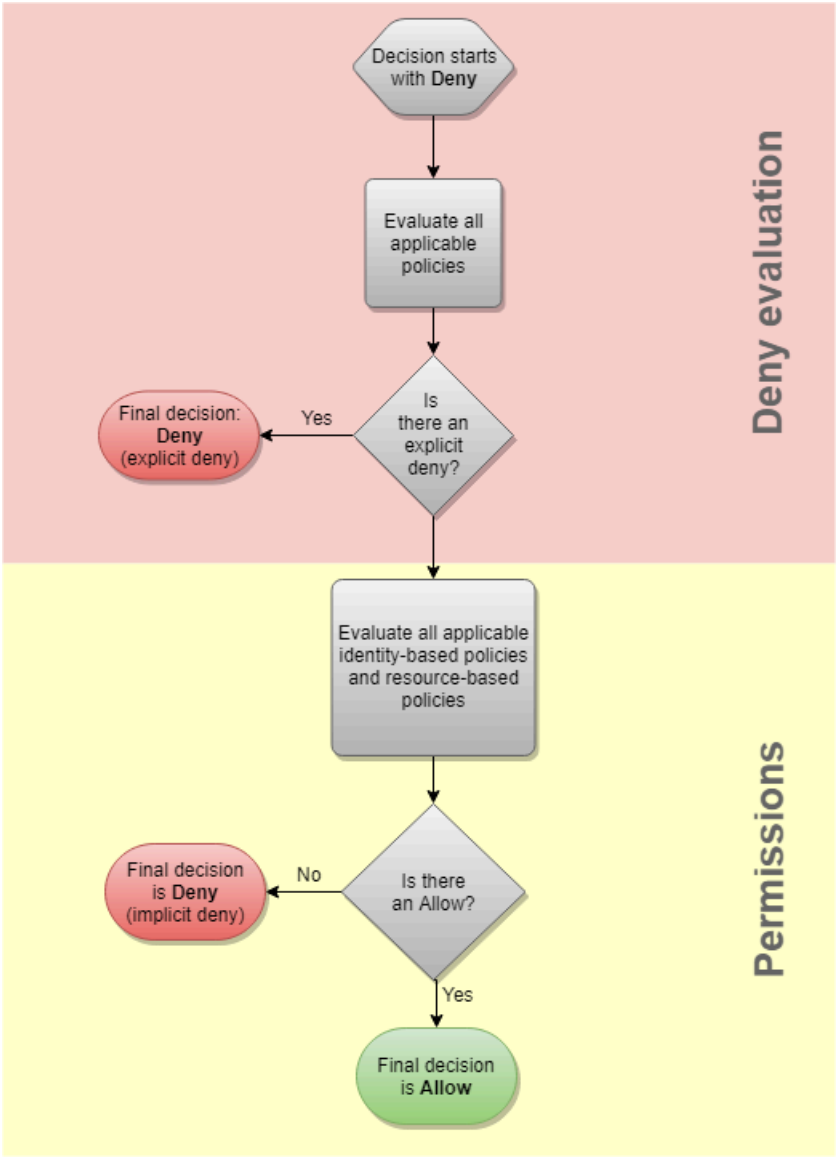
此外，以下基于资源的策略（称为存储桶策略）附加到 amzn-s3-demo-bucket-carlossalazar 存储桶。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::123456789012:user/carlossalazar"
            },
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::amzn-s3-demo-bucket-carlossalazar/*",
                "arn:aws:s3:::amzn-s3-demo-bucket-carlossalazar"
            ]
        }
    ]
}
```

此策略指定只有 carlossalazar 用户可以访问 amzn-s3-demo-bucket-carlossalazar 存储桶。

当 Carlos 请求将文件保存到 amzn-s3-demo-bucket-carlossalazar-logs 存储桶时，Amazon 将确定应用于请求的策略。在此示例中，仅基于身份的策略和基于资源的策略适用。这些都是权限策略。由于没有任何权限边界适用，评估逻辑将减少为以下逻辑。



Amazon 首先检查应用于请求上下文的 Deny 语句。它找到一个，因为基于身份的策略显式拒绝 Carlos 访问用于日志记录的任何 S3 存储桶。拒绝 Carlos 访问。

假设他随后意识到自己的错误，尝试将文件保存到 amzn-s3-demo-bucket-carlossalazar 存储桶。Amazon 检查 Deny 语句，但未找到。然后，它检查权限策略。基于身份的策略和基于资源的策略均允许请求。因此，Amazon 允许请求。如果其中任何一个显式拒绝了语句，则将拒绝请求。如果其中一种策略类型允许请求，而另一种不允许，则仍允许请求。

跨账户策略评估逻辑

您可以允许一个账户中的主体访问另一个账户中的资源。这称为 cross-account access（跨账户访问）。当您允许跨账户访问时，主体所在的账户称为受信任 账户。资源所在的账户是信任 账户。

要允许跨账户访问，请将基于资源的策略附加到您要共享的资源。您还必须向在请求中充当主体的身份附加基于身份的策略。信任账户中基于资源的策略必须指定受信任账户中有权访问资源的主体。您可以指定整个账户或其 IAM 用户、Amazon STS 联合用户主体、IAM 角色或代入角色会话。您还可以将 Amazon 服务指定为主体。有关更多信息，请参阅 如何指定主体。

主体的基于身份的策略必须允许对信任服务中的资源进行请求的访问。您可以通过指定资源的 ARN 来执行此操作。

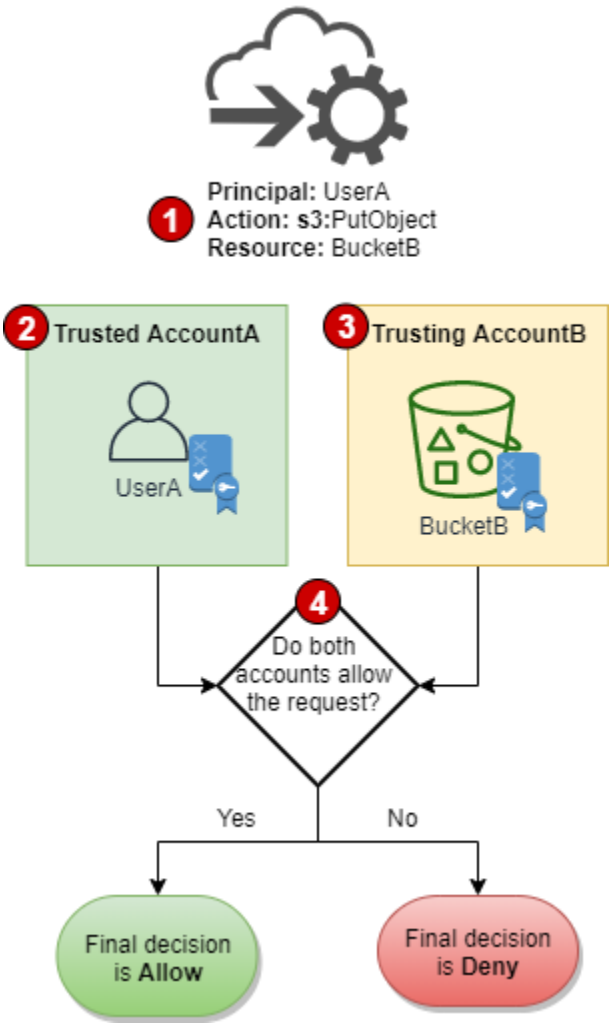
在 IAM 中，您可以将基于资源的策略附加到 IAM 角色，以允许其他账户中的主体代入该角色。角色的基于资源的策略称为角色信任策略。代入该角色之后，允许的主体可以使用生成的临时凭证访问您账户中的多个资源。此访问权限在角色的基于身份的权限策略中定义。如需了解使用角色允许跨账户访问与使用其他基于资源的策略允许跨账户访问之间的不同之处，请参阅IAM 中的跨账户资源访问。

> **Important**
>
> 其他服务可能会影响策略评估逻辑。例如，Amazon Organizations 支持可应用于一个或多个账户中的主体和资源的服务控制策略和资源控制策略。Amazon Resource Access Manager 支持策略片段，用于控制允许主体对与其共享的资源执行哪些操作。

确定是否允许跨账户请求

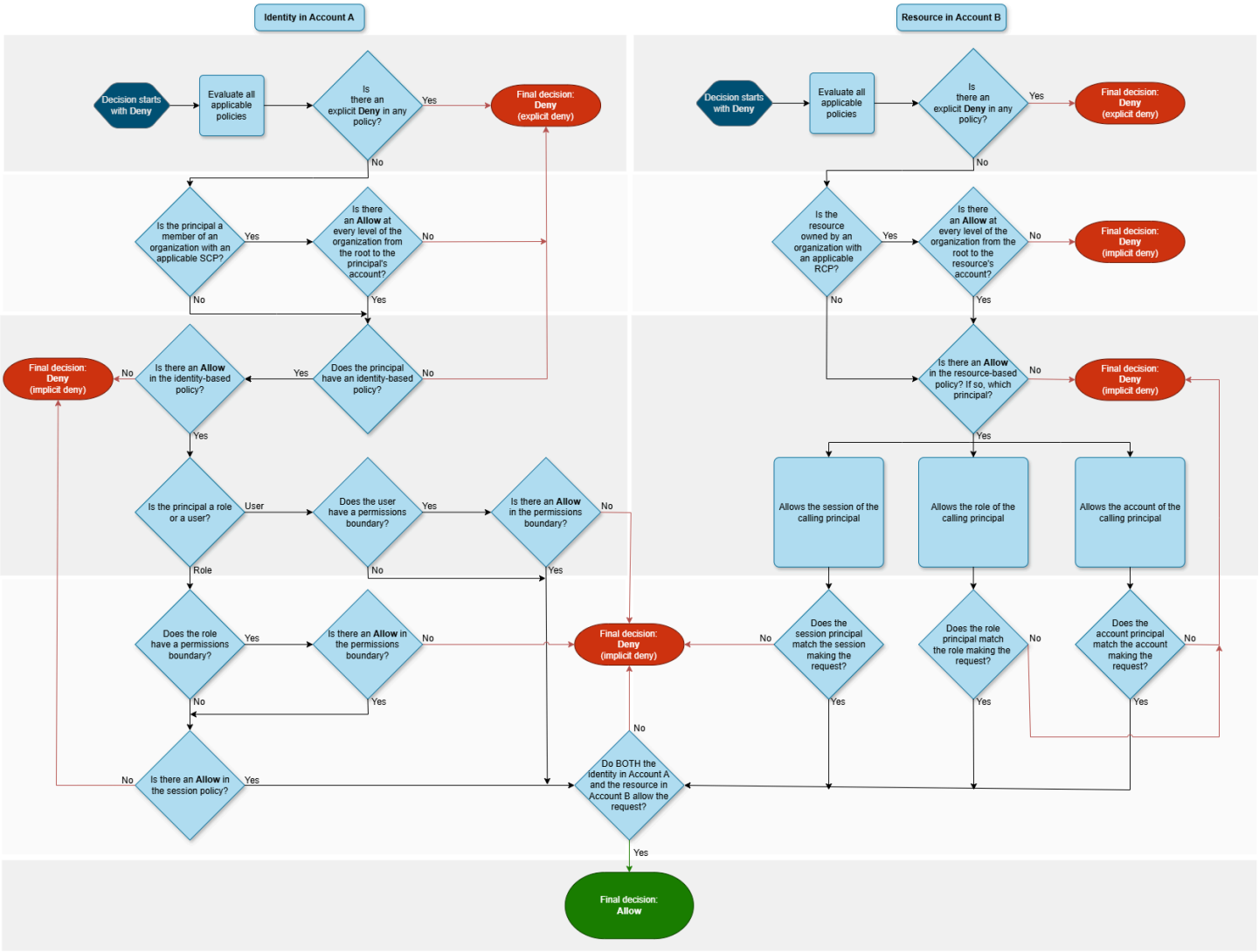
对于跨账户请求，受信任账户 `AccountA` 中的请求者必须具有基于身份的策略。该策略必须允许他们向信任账户 `AccountB` 中的资源发出请求。此外，`AccountB` 中的基于资源的策略必须允许 `AccountA` 中的请求者访问资源。

当您发出跨账户请求时，Amazon 会执行两个评估。Amazon 评估信任账户和受信任账户中的请求。有关如何在单个账户中评估请求的更多信息，请参阅Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。仅当两个评估都返回 `Allow` 决策时，才允许该请求。



1. 当一个账户中的主体发出请求以访问另一个账户中的资源时，这是一个跨账户请求。
2. 请求主体存在于受信任账户 (AccountA) 中。当 Amazon 评估此账户时，它会检查基于身份的策略以及可以限制基于身份的策略的任何策略。有关更多信息，请参阅 评估具有权限边界的基于身份的策略。
3. 请求的资源存在于信任账户 (AccountB) 中。当 Amazon 评估此账户时，它会检查附加到所请求资源的基于资源的策略，以及可以限制基于资源的策略的任何策略。有关更多信息，请参阅 评估基于身份的策略以及基于资源的策略。
4. 仅当两个账户策略评估均允许该请求时，Amazon 才允许该请求。

以下流程图详细介绍了如何针对跨账户请求作出策略评估决策。同样，仅当两个账户策略评估均允许该请求时，Amazon 才会允许该请求。



跨账户策略评估示例

以下示例演示了一个账户中基于资源的策略向另一个账户中的角色授予权限的情况。

假设 Carlos 是一名开发人员，在账户 111111111111 中具有名为 Demo 的 IAM 角色。他想要将文件保存到账户 222222222222 中的 `amzn-s3-demo-bucket-production-logs` Amazon S3 存储桶。

还假定将以下策略附加到 Demo IAM 角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
```

```
        {
            "Sid": "AllowS3ListRead",
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "*"
        },
        {
            "Sid": "AllowS3ProductionObjectActions",
            "Effect": "Allow",
            "Action": "s3:*Object*",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-production/*"
        },
        {
            "Sid": "DenyS3Logs",
            "Effect": "Deny",
            "Action": "s3:*",
            "Resource": [
                "arn:aws:s3:::*log*",
                "arn:aws:s3:::*log*/*"
            ]
        }
    ]
}
```

此策略中的 AllowS3ListRead 语句允许 Carlos 查看 Amazon S3 中所有存储桶的列表。AllowS3ProductionObjectActions 语句允许 Carlos 对 amzn-s3-demo-bucket-production 存储桶中对象的完全访问权限。

此外，以下基于资源的策略（称为存储桶策略）附加到账户 222222222222 中的 amzn-s3-demo-bucket-production 存储桶。

JSON

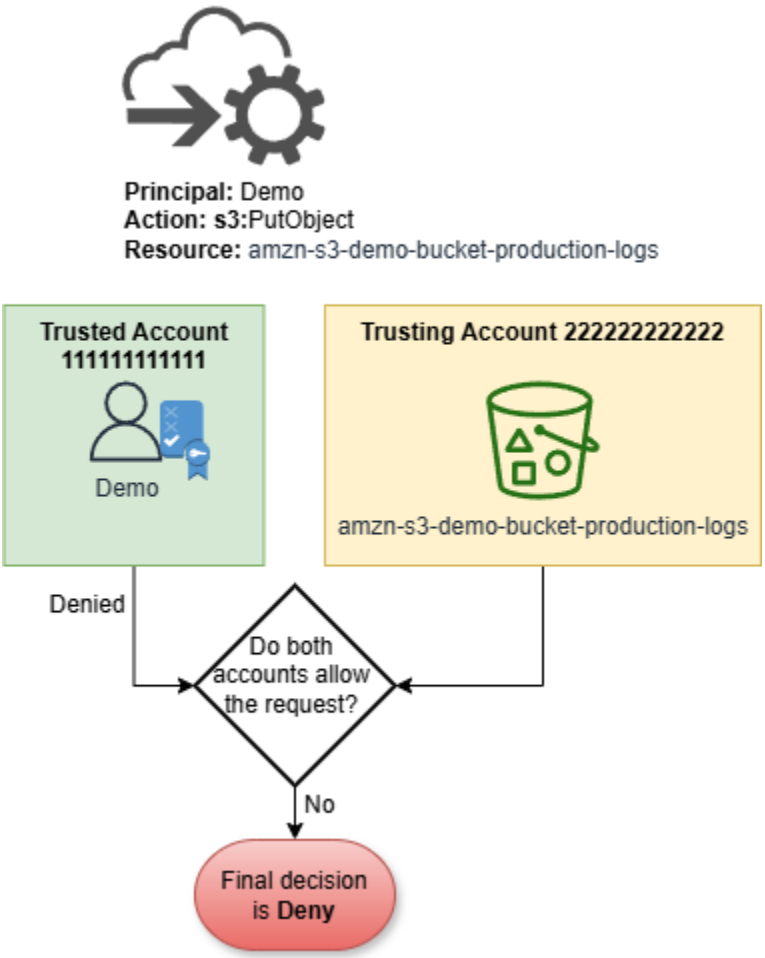
```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "s3:GetObject*",
                "s3:PutObject*",
```

```
                "s3:ReplicateObject",
                "s3:RestoreObject"
            ],
            "Principal": { "AWS": "arn:aws:iam::111111111111:role/Demo" },
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket-production/*"
        }
    ]
}
```

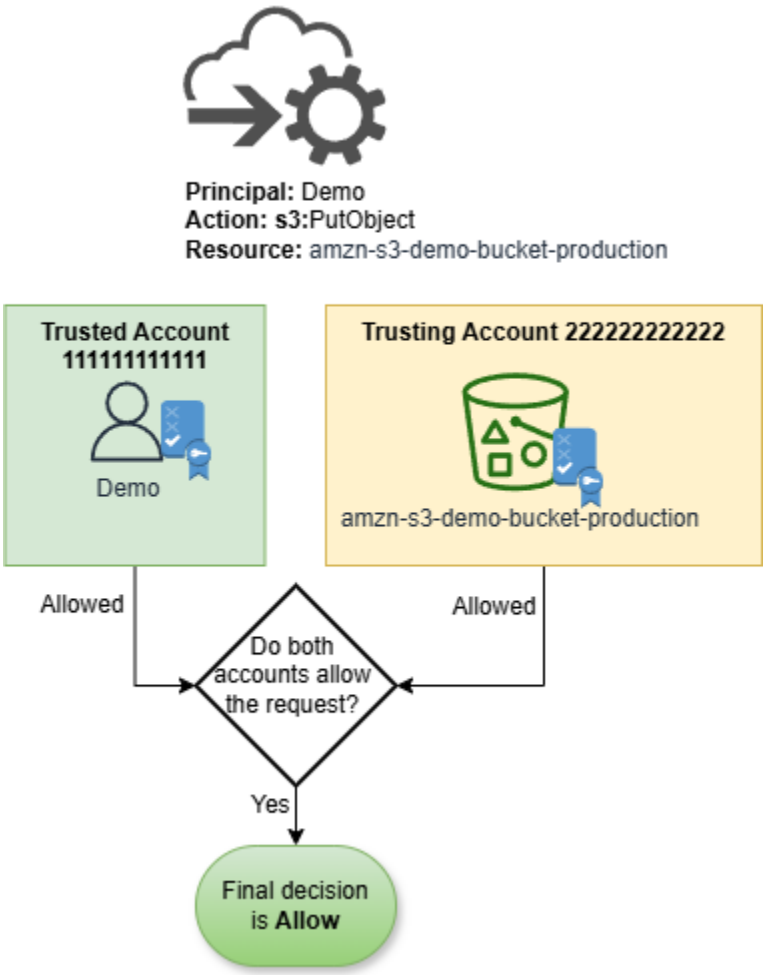
此策略允许 Demo 角色访问 amzn-s3-demo-bucket-production 存储桶中的对象。该角色以创建和编辑存储桶中的对象，但不能删除。该角色无法管理存储桶本身。

当 Carlos 请求将文件保存到 amzn-s3-demo-bucket-production-logs 存储桶时，Amazon 将确定应用于请求的策略。在这种情况下，附加到 Demo 角色的基于身份的策略是在账户 111111111111 中应用的唯一策略。在账户 222222222222 中，amzn-s3-demo-bucket-production-logs 存储桶没有附加基于资源的策略。Amazon 评估账户 111111111111 时返回了决策 Deny。这是因为基于身份的策略中的 DenyS3Logs 语句明确拒绝访问任何日志存储桶。有关如何在单个账户中评估请求的更多信息，请参阅Amazon 执行代码逻辑如何评估允许或拒绝访问的请求。

由于在一个账户中明确拒绝了请求，因此最终决策是拒绝请求。



假设随后 Carlos 意识到他的错误并尝试将文件保存到 `Production` 存储桶。Amazon 首先检查账户 111111111111 以确定是否允许请求。仅基于身份的策略适用并允许请求。随后，Amazon 检查账户 222222222222。仅附加到 `Production` 存储桶的基于资源的策略适用，并允许请求。由于两个账户均允许请求，因此最终决策是允许请求。



显式拒绝和隐式拒绝之间的区别

如果适用策略包含 Deny 语句，则请求会导致显式拒绝。如果应用于请求的策略包含一个 Allow 语句和一个 Deny 语句，Deny 语句优先于 Allow 语句。将显式拒绝请求。

当没有适用的 Deny 语句但也没有适用的 Allow 语句时，会发生隐式拒绝。由于原定设置下拒绝访问 IAM 主体，因此必须显式允许他们执行操作。否则，将隐式拒绝他们访问。

在设计授权战略时，您必须创建包含 Allow 语句的策略才能允许主体成功发出请求。但是，您可以选择显式拒绝和隐式拒绝的任意组合。

例如，您可以创建以下策略，其中包括允许的操作、隐式拒绝的操作和显式拒绝的操作。AllowGetList 语句允许对 IAM 操作的只读权限，这些操作以 Get 和 List 为前缀。IAM 中的所有其他操作，例如 iam:CreatePolicy 是隐式拒绝。DenyReports 语句通过拒绝对包含 Report 后缀（例如 iam:GetOrganizationsAccessReport）的操作的访问权限，显式拒绝对 IAM 报告的访问权限。如果有人向此主体添加另一个策略以授予他们访问 IAM 报告的权限（例如 iam:GenerateCredentialReport），由于这种明确拒绝，报告相关请求仍被拒绝。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowGetList",
            "Effect": "Allow",
            "Action": [
                "iam:Get*",
                "iam:List*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "DenyReports",
            "Effect": "Deny",
            "Action": "iam:*Report",
            "Resource": "*"
        }
    ]
}
```

IAM JSON 策略语言的语法

该页提供了用于在 IAM 中创建 JSON 策略的语言的正式语法。提供此语法的目的是为了便于您理解如何构造和验证策略。

有关策略示例，请参阅以下主题：

- Amazon Identity and Access Management 中的策略和权限
- IAM 基于身份的策略示例
- 《Amazon EC2 用户指南》中的用于 Amazon EC2 控制台的策略示例和使用 Amazon CLI、Amazon EC2 CLI 或 Amazon SDK 的策略示例。
- Amazon Simple Storage Service User Guide（亚马逊云科技 Simple Storage Service 用户指南）中的存储桶策略示例和用户策略示例。

有关其他 Amazon 产品中使用的策略的示例，请参阅相应服务的文档。

主题

- 策略语言和 JSON
- 此语法采用的约定
- 语法
- 策略语法说明

策略语言和 JSON

策略是以 JSON 格式表示的。当您创建或编辑 JSON 策略时，IAM 可以执行策略验证以帮助您创建有效的策略。IAM 可识别 JSON 语法错误，而 IAM Access Analyzer 将提供额外的策略检查和建议，以帮助您进一步优化策略。要了解策略验证的更多信息，请参阅 IAM 策略验证。要了解有关 IAM Access Analyzer 策略检查和可执行建议的更多信息，请参阅 IAM Access Analyzer 策略验证。

在此文档中，我们不提供有效 JSON 组成部分的完整说明。而是提供一些基本的 JSON 规则：

- 各实体间允许使用空格。
- 值用引号括起来。对于数字值和布尔值，引号是可选的。
- 很多元素 (例如 `action_string_list` 和 `resource_string_list`) 都可以采用 JSON 数组作为值。数组可以采用一个或多个值。如果包含多个值，则数组用方括号 ([和]) 括起来并用逗号分隔，如以下示例中所示：

  ```
  "Action" : ["ec2:Describe*","ec2:List*"]
  ```
- 基本 JSON 数据类型 (布尔型、数字和字符串) 在 RFC 7159 中定义。

此语法采用的约定

此语法采用以下约定：

- 以下字符是 JSON 令牌，包含 在策略中：

  ```
  { } [ ] " , :
  ```
- 以下字符是语法中的特殊字符，不包含 在策略中：

  ```
  = < > ( ) |
  ```
- 当一个元素允很多个值时，使用重复值、逗号分隔符和省略号 (`...`) 进行表示。示例：

  ```
  [<action_string>, <action_string>, ...]
  ```

```
<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }
```

允许多个值时，只包含一个值也是有效的。只有一个值时，必须省略尾部的逗号。如果元素采用数组 (使用 [和] 标记) 但只包含一个值，则括号可选。示例：

```
"Action": [<action_string>]
```

```
"Action": <action_string>
```

- 元素后的问号 (?) 表示该元素是可选的。示例：

```
<version_block?>
```

不过，请务必参考语法列表后的说明了解有关可选元素的详细信息。

- 元素之间的竖线 (|) 表示备选项。在语法中，圆括号定义备选项的范围。示例：

```
("Principal" | "NotPrincipal")
```

- 必须为文字字符串的元素括在双引号 (") 中。示例：

```
<version_block> = "Version" : ("2008-10-17" | "2012-10-17" )
```

有关更多说明，请参阅语法列表后面的策略语法说明。

语法

下面的列表描述了策略语言语法。有关此列表中采用的约定，请参阅前面的部分。有关更多信息，请参阅后面的说明。

> **Note**
>
> 此语法描述了使用版本号 2008-10-17 和 2012-10-17 标记的策略。Version 策略元素与策略版本不同。Version 策略元素用在策略之中，用于定义策略语言的版本。另一方面，当您对 IAM 中的客户托管策略进行更改时，将创建一个策略版本。已更改的策略不会覆盖现有策略。而是由 IAM 创建新的托管策略版本。要了解 Version 策略元素的更多信息，请参阅IAM JSON 策略元素：Version。要了解策略版本的更多信息，请参阅the section called “IAM 策略版本控制”。

```
policy  = {
```

```
     <version_block?>,
     <id_block?>,
     <statement_block>
}

<version_block> = "Version" : ("2008-10-17"        | "2012-10-17"       )

<id_block> = "Id" : <policy_id_string>

<statement_block> = "Statement" : [ <statement>, <statement>, ... ]

<statement> = {
    <sid_block?>,
    <principal_block?>,
    <effect_block>,
    <action_block>,
    <resource_block>,
    <condition_block?>
}

<sid_block> = "Sid" : <sid_string>

<effect_block> = "Effect" : ("Allow" | "Deny")

<principal_block> = ("Principal" | "NotPrincipal") : ("*" | <principal_map>)

<principal_map> = { <principal_map_entry>, <principal_map_entry>, ... }

<principal_map_entry> = ("AWS" | "Federated" | "Service" | "CanonicalUser") :
    [<principal_id_string>, <principal_id_string>, ...]

<action_block> = ("Action" | "NotAction") :
    ("*" | <action_string> | [<action_string>, <action_string>, ...])

<resource_block> = ("Resource" | "NotResource") :
    : ("*" | <resource_string> | [<resource_string>, <resource_string>, ...])

<condition_block> = "Condition" : { <condition_map> }
<condition_map> = {
  <condition_type_string> : { <condition_key_string> : <condition_value_list> },
  <condition_type_string> : { <condition_key_string> : <condition_value_list> }, ...
}
<condition_value_list> = [<condition_value>, <condition_value>, ...]
```

```
<condition_value> = (<condition_value_string> | <condition_value_string> |
 <condition_value_string>)
```

策略语法说明

- 一个策略可以包含一组语句。
- 策略的最大大小介于 2048 个字符和 10,240 个字符之间，具体取决于策略所附加到的实体。有关更多信息，请参阅 IAM 和 Amazon STS 配额。策略大小计算不包括空格字符。
- 单个元素不能包含同一个键的多个实例。例如，不能在同一语句中包含两次 `Effect` 块。
- 各个块的显示顺序无限制。例如，在策略中，`version_block` 可以跟在 `id_block` 后面。同样地，`effect_block`、`principal_block`、`action_block` 在语句中也可以按任何顺序显示。
- `id_block` 在基于资源的策略中是可选的。它一定不能 包含在基于身份的策略中。
- `principal_block` 元素在基于资源的策略中（例如，在 Amazon S3 存储桶策略中）和 IAM 角色的信任策略中是必需的。它一定不能 包含在基于身份的策略中。
- Amazon S3 存储桶策略中的 `principal_map` 元素可能包含 `CanonicalUser` ID。大多数基于资源的策略不支持该映射。要了解有关在存储桶策略中使用规范用户 ID 的更多信息，请参阅 Amazon Simple Storage Service 用户指南中的在策略中指定主体。
- 每个字符串值 (`policy_id_string`、`sid_string`、`principal_id_string`、`action_string`、`resource_stri` 以及 `condition_value` 的字符串版本) 都可以有其自己的最小和最大长度限制、具体的允许值或必需的内部格式。

有关字符串值的说明

这一部分提供了有关在策略的不同元素中使用的字符串值的更多信息。

`action_string`

由服务命名空间、冒号和操作名称组成。操作名称可以包含通配符。示例：

```
"Action":"ec2:StartInstances"

"Action":[
  "ec2:StartInstances",
  "ec2:StopInstances"
]

"Action":"cloudformation:*"
```

```
"Action":"*"

"Action":[
  "s3:Get*",
  "s3:List*"
]
```

policy_id_string

提供一种方法将关于策略的信息作为整体包含在其中。某些服务 (如 Amazon SQS 和 Amazon SNS) 以预留方式使用 Id 元素。除非另受某个服务限制，否则 policy_id_string 可以包含空格。有些服务要求此值在 Amazon 账户内唯一。

> **Note**
>
> id_block 可在基于资源的策略中使用，但不能在基于身份的策略中使用。

长度没有限制，但此字符串计入策略总长度，而策略总长度是有限制的。

```
"Id":"Admin_Policy"

"Id":"cd3ad3d9-2776-4ef1-a904-4c229d1642ee"
```

sid_string

提供一种方法来包含有关单个语句的信息。对于 IAM policy，Sid 值中仅允许使用基本的字母数字（A-Z、a-z、0-9）字符。支持资源策略的其他 Amazon 服务对于 Sid 值可能有其他要求。例如，一些服务要求此值在 Amazon Web Services 账户 内唯一，而一些服务允许在 Sid 值中使用其他字符，如空格。

```
"Sid":"1"

"Sid": "ThisStatementProvidesPermissionsForConsoleAccess"
```

principal_id_string

提供使用 Amazon Web Services 账户、IAM 用户、IAM 角色、联合用户或代入角色的用户的 Amazon 资源名称（ARN） 指定主体的方式。对于 Amazon Web Services 账户，您还可以使用简

单形式 AWS:*accountnumber* 而不是完整的 ARN。有关所有选项（包括 Amazon 服务、担任的角色等），请参阅如何指定主体。

请注意，您只能使用 * 指定“everyone/anonymous”。您不能使用它指定一部分名称或 ARN。

resource_string

在大多数情况下，由一个 Amazon Resource Name (ARN) 组成。您可以在 ARN 的资源部分使用通配符（* 和 ?）。有关在 ARN 中使用通配符的更多信息，请参阅 在路径中使用通配符。

> **Note**
>
> 当您在基于身份的策略中指定不完整的 ARN（少于标准的六个字段）时，Amazon 会通过在所有缺失字段中添加通配符 (*) 来自动完成 ARN。例如，指定 `arn:aws:sqs` 相当于 `arn:aws:sqs:*:*:*`，它让您可以访问所有区域和账户中的所有 Amazon SQS 资源。但是，传递给 Amazon STS AssumeRole、AssumeRoleWithWebIdentity 和 AssumeRoleWithSAML 请求的会话策略不支持不完整的 ARN。在会话策略中使用不完整的 ARN 会导致 `MalformedPolicyDocumentException` 错误。

```
"Resource":"arn:aws:iam::123456789012:user/Bob"

"Resource":"arn:aws:s3:::amzn-s3-demo-bucket/*"
```

condition_type_string

标识所测试条件的类型，例如 StringEquals、StringLike、NumericLessThan、DateGreaterThanEquals、Bool、BinaryE 等。有关条件类型的完整列表，请参阅IAM JSON 策略元素：条件运算符。

```
"Condition": {
  "NumericLessThanEquals": {
    "s3:max-keys": "10"
  }
}

"Condition": {
  "Bool": {
    "aws:SecureTransport": "true"
  }
}
```

```
"Condition": {
  "StringEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}
```

condition_key_string

标识将对其值进行测试以便确定条件是否满足的条件键。Amazon 定义了一组在所有 Amazon 服务（包括 aws:PrincipalType、aws:SecureTransport 和 aws:userid）中可用的条件键。

有关 Amazon 条件键的列表，请参阅 Amazon 全局条件上下文密钥。有关服务特定的条件键，请参阅相应服务的文档，例如：

- Amazon Simple Storage Service 用户指南中的在策略中指定条件
- 《Amazon EC2 用户指南》中的 Amazon EC2 的 IAM 策略。

```
"Condition":{
  "Bool": {
      "aws:SecureTransport": "true"
   }
}

"Condition": {
  "StringNotEquals": {
      "s3:x-amz-server-side-encryption": "AES256"
   }
}

"Condition": {
  "StringEquals": {
    "aws:ResourceTag/purpose": "test"
  }
}
```

condition_value_string

标识 condition_key_string 的值，该值决定了是否满足条件。有关条件类型的有效值的完整列表，请参阅 IAM JSON 策略元素：条件运算符。

```
"Condition":{
```

```
  "ForAnyValue:StringEquals": {
  "dynamodb:Attributes": [
   "ID",
   "PostDateTime"
         ]
  }
}
```

工作职能的 Amazon 托管策略

我们建议使用授予最低权限的策略，或仅授予执行任务所需的许可。授予最小权限的最安全方式是编写一个仅具有团队所需权限的自定义策略。您必须创建一个流程，以允许您的团队在必要时请求更多权限。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。

要开始向您的 IAM 身份（用户、用户组和角色）添加权限，您可以使用 Amazon托管策略。Amazon 托管策略涵盖常见使用案例，并且可在您的 Amazon Web Services 账户 中使用。Amazon 托管策略不会授予最低权限。您必须考虑授予您的主体超出其完成工作所需的更多权限所带来的安全风险。

您可以将 Amazon 托管策略（包括任务函数）附加到任何 IAM 身份。要切换到最低权限，您可以运行 Amazon Identity and Access Management and Access Analyzer 以使用 Amazon 托管式策略监控主体。了解他们使用的权限后，您可以编写自定义策略或生成仅包含团队所需权限的策略。这中方法不太安全，但您能够以更灵活的方式了解您的团队如何使用 Amazon。

Amazon 工作职能托管策略，旨在贴合 IT 行业的常见工作职能。您可以使用这些策略，给特定工作职能的某人，授予执行预期任务所需的权限。这些策略将许多服务的权限整合到单个策略，比起分散在多个策略之间的权限，处理起来更加简单。

使用角色来合并服务

某些策略使用 IAM 服务角色帮助您利用在其他 Amazon 服务中发现的功能。这些策略授予对 `iam:passrole` 的访问权限，允许使用策略的用户将角色传递给 Amazon 服务。此角色向 Amazon 服务委派代表您执行操作的 IAM 权限。

您必须根据需求创建角色。例如，网络管理员策略允许具有策略的用户向 Amazon CloudWatch 服务传递名为 “flow-logs-vpc”的角色。CloudWatch 使用该角色记录和捕获用户创建的 VPC 的 IP 流量。

要遵循安全最佳实践，工作职能的策略包括限制可传递的有效角色名称的筛选器。这有助于避免授予不必要的权限。如果您的用户确实需要可选的服务角色，您必须按照策略中指定的命名约定创建一个角色。随后才能给角色授予权限。完成该操作后，用户方可配置服务来使用该角色，为服务授予角色提供的任何权限。

在以下部分中，每个策略的名称都是指向 Amazon Web Services 管理控制台中策略详细信息页面的链接。您可以在该页面中查看策略文档并了解它所授予的权限。

管理员任务函数

Amazon 托管策略名称：AdministratorAccess

使用案例：此用户具有完全访问权限，而且可以为 Amazon; 中的每个服务和资源委派权限。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略为账户中的所有 Amazon 服务和所有资源授予所有操作权限。有关托管式策略的更多信息，请参阅《Amazon 托管策略参考指南》中的 AdministratorAccess。

账单任务函数

Amazon 托管策略名称：Billing

使用案例：此用户需要查看账单信息、设置付款和授权付款。用户可以监控整个 Amazon 服务累积的各项成本。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略授予管理账单、成本、付款方式、预算和报告的完整权限。有关其他成本管理策略示例，请参阅《Amazon 账单与成本管理 用户指南》中的Amazon Billing 策略示例。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 Billing。

数据库管理员任务函数

Amazon 托管策略名称：DatabaseAdministrator

使用案例：此用户设置、配置和维护 Amazon 云中的数据库。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略说明：此策略授予创建、配置和维护数据库的权限。它包括访问 Amazon 数据库服务（例如 Amazon DynamoDB、Amazon Relational Database Service (RDS) 和 Amazon Redshift）。有关此策

略支持的数据库服务的完整列表，请查看此策略。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 DatabaseAdministrator。

此作业功能策略支持将角色传递给 Amazon 服务的功能。该策略仅允许对下表中指定的角色执行 `iam:PassRole` 操作。有关更多信息，请参阅本主题后面的创建角色并附加策略（控制台）。

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	选择此 Amazon 托管策略
允许用户监控 RDS 数据库	rds-monitoring-role	Amazon RDS 角色用于增强监控	AmazonRDSEnhancedMonitoringRole
允许 Amazon Lambda 监控您的数据库和访问外部数据库	rdbms-lambda-access	Amazon EC2	AWSLambda_FullAccess
允许 Lambda 使用 DynamoDB 将文件上传到 Amazon S3 和 Amazon Redshift 集群	lambda_exec_role	Amazon Lambda	按照 Amazon 大数据博客中的定义，新建一个托管策略
允许 Lambda 函数触发您的 DynamoDB 表	lambda-dynamodb-*	Amazon Lambda	AWSLambdaDynamoDBExecutionRole
允许 Lambda 函数访问 VPC 中的 Amazon RDS	lambda-vpc-execution-role	使用 Amazon Lambda 开发人员指南中定义的信任策略创建一个角色	AWSLambdaVPCAccessExecutionRole
允许 Amazon Data Pipeline 访问您的 Amazon 资源	DataPipelineDefaultRole	使用 Amazon Data Pipeline 开发人员指南中定义的信任策略创建一个角色	Amazon Data Pipeline 文档列出了此使用案例所需的权限。请参阅适用于 Amazon Data Pipeline 的 IAM 角色

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	选择此 Amazon 托管策略
允许您在 Amazon EC2 实例上运行的应用程序访问您的 Amazon 资源	DataPipelineDefaultResourceRole	使用 Amazon Data Pipeline 开发人员指南中定义的信任策略创建一个角色	AmazonEC2RoleforDataPipelineRole

数据科学家任务函数

Amazon 托管策略名称：DataScientist

使用案例： 此用户运行 Hadoop 作业和查询。该用户还访问并分析有关数据分析和商业智能的信息。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略授予在 Amazon EMR 集群上创建、管理和运行查询的权限，以及使用 Amazon QuickSight 等工具执行数据分析的权限。该策略包括访问其他数据科学家服务，例如 Amazon Data Pipeline、Amazon EC2、Amazon Kinesis、Amazon Machine Learning 和 SageMaker AI。有关此策略支持的数据科学家服务的完整列表，请查看此策略。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 DataScientist。

此作业功能策略支持将角色传递给 Amazon 服务的功能。一个语句允许将任何角色传递给 SageMaker AI。另一个语句仅允许对下表中指定的角色执行 `iam:PassRole` 操作。有关更多信息，请参阅本主题后面的创建角色并附加策略（控制台）。

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon EC2 实例访问适合集群的服务和资源	EMR-EC2_DefaultRole	Amazon EMR for EC2	AmazonElasticMapReduceforEC2Role

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon EMR 访问适合集群的 Amazon EC2 服务和资源	EMR_DefaultRole	Amazon EMR	AmazonEMR ServicePolicy_v2
允许适用于 Apache Flink 的 Kinesis 托管服务访问串流数据来源	kinesis-*	使用Amazon 大数据博客中定义的信任策略创建一个角色。	请参阅 Amazon; 大数据博客，其中根据您的使用情况概述了四种可能的选项。
允许 Amazon Data Pipeline 访问您的 Amazon 资源	DataPipelineDefaultRole	使用 Amazon Data Pipeline 开发人员指南中定义的信任策略创建一个角色	Amazon Data Pipeline 文档列出了此使用案例所需的权限。请参阅适用于 Amazon Data Pipeline 的 IAM 角色
允许您在 Amazon EC2 实例上运行的应用程序访问您的 Amazon 资源	DataPipelineDefaultResourceRole	使用 Amazon Data Pipeline 开发人员指南中定义的信任策略创建一个角色	AmazonEC2 RoleforDataPipelineRole

高级开发人员用户任务函数

Amazon 托管策略名称：PowerUserAccess

使用案例：此用户执行应用程序开发任务，并可创建和配置特定的资源和服务，它们为能够识别 Amazon 的应用程序开发提供支持。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略的第一条语句使用 NotAction 元素以允许对于所有 Amazon 服务和所有资源执行所有操作（Amazon Identity and Access Management、Amazon Organizations 和 Amazon 账户管理除外）。第二条语句授予创建服务相关角色的 IAM 权限。这是某些服务访问其他服务（如 Amazon S3 存储桶）中的资源所必需的。该策略还授予 Amazon Organizations 权限以查看有关用户所在组织的信息，包括管理账户电子邮件和企业限制。尽管此策略限制 IAM、Amazon Organizations，但它允许用户执行所有 IAM Identity Center 操作（如果 IAM Identity Center 已启用）。它还授予账户管理权限以查看为账户启用或禁用了哪些 Amazon 区域。

网络管理员任务函数

Amazon托管策略名称：NetworkAdministrator

使用案例：此用户分配到的任务是设置和维护 Amazon 网络资源。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略说明：此策略授予在 Auto Scaling、Amazon EC2、Amazon Direct Connect、Route 53、Amazon CloudFront、Elastic Load Balancing、Amazon Elastic Beanstalk、Amazon SNS、CloudWatch、CloudWatch Logs、Amazon S3、IAM 以及 Amazon Virtual Private Cloud 中创建和维护网络资源的权限。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 NetworkAdministrator。

此工作职能需要将角色传递给 Amazon 服务的能力。该策略仅为下表中列出的角色授予 `iam:GetRole` 和 `iam:PassRole` 的权限。有关更多信息，请参阅本主题后面的创建角色并附加策略（控制台）。

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon VPC 代表用户在 CloudWatch Logs 中创建和管理日志，以监控进出您的 VPC 的 IP 流量	flow-logs-*	使用 Amazon VPC 用户指南中定义的信任策略创建一个角色	此使用案例没有现成的 Amazon 托管策略，但是，文档列出了所需权限。参阅 Amazon VPC 用户指南。

只读访问权限

Amazon 托管策略名称： ReadOnlyAccess

使用案例：此用户需要对 Amazon Web Services 账户 中所有资源的只读访问权限。

> **Important**
>
> 该用户还可以读取 Amazon S3 存储桶和 Amazon DynamoDB 表等存储服务中的数据。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略说明：此策略授予列出、获取、描述和查看资源及其属性的权限。它不包括创建或删除等转换功能。此策略包含对与安全相关的 Amazon 服务（例如 Amazon Identity and Access Management 和 Amazon 账单与成本管理）的只读访问权限。有关此策略支持的服务和操作的完整列表，请查看此策略。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 ReadOnlyAccess。如果需要不授予存储服务中数据读取权限的类似策略，请参阅 “仅查看”用户任务函数。

MCP 服务操作完全访问权限

Amazon 托管策略名称：AWSMcpServiceActionsFullAccess

使用案例：此用户需要使用 Amazon MCP 服务器才能访问 Amazon 服务。此策略不向其他 Amazon 服务授予对 MCP 服务所采取操作的访问权限。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略授予调用任何 Amazon MCP 服务操作的权限。如果您无需为每个 Amazon MCP 服务指定权限，可以使用此策略。其不会向其他 Amazon 服务授予 MCP 服务所执行操作的权限，这些权限必须始终单独授予，且在 MCP 服务操作权限之外授予。有关托管式策略的更多信息，请参阅《Amazon Managed Policy Reference Guide》中的 AWSMcpServiceActionsFullAccess。

安全审计员任务函数

Amazon 托管策略名称：SecurityAudit

使用案例： 此用户监控账户是否符合安全要求。此用户可以访问日志和事件来调查潜在的安全漏洞或潜在的恶意活动。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略可授予查看多项 Amazon 服务的配置数据并检查其日志的权限。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 SecurityAudit。

支持用户任务函数

Amazon 托管式策略名称： AWSSupportAccess

使用案例：此用户与 Amazon Support 联系，创建支持案例并查看现有案例的状态。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略授予创建和更新 Amazon Web Services 支持 工单的权限。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 AWSSupportAccess。

系统管理员任务函数

Amazon 托管策略名称：SystemAdministrator

使用案例： 此用户设置和维护用于开发操作的资源。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略说明：此策略授予权限，以跨诸多 Amazon 服务（包括 Amazon CloudTrail、Amazon CloudWatch、Amazon CodeCommit、Amazon CodeDeploy、Amazon Config、Amazon Directory Service、Amazon EC2、Amazon Identity and Access Management、Amazon Key Management Service、Amazon Lambda、Amazon RDS、Route 53、Amazon S3、Amazon SES、Amazon SQS、Amazon Trusted Advisor 和 Amazon VPC）创建和维护资源。有关托管式策略的更多信息，请参阅《Amazon 托管策略参考指南》中的 SystemAdministrator。

此工作职能需要将角色传递给 Amazon 服务的能力。该策略仅为下表中列出的角色授予 `iam:GetRole` 和 `iam:PassRole` 的权限。有关更多信息，请参阅本主题后面的创建角色并附加策略（控制台）。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

使用案例	角色名称 (* 为通配符)	可选择的服务角色类型	可选择的 Amazon 托管策略
允许 Amazon ECS 集群中在 EC2 实例中运行的应用程序访问 Amazon ECS	ecr-sysadmin-*	适合 EC2 Container Service 的 Amazon EC2 角色	AmazonEC2ContainerServiceforEC2Role
允许用户监控数据库	rds-monitoring-role	Amazon RDS 角色用于增强监控	AmazonRDSEnhancedMonitoringRole
允许在 EC2 实例中运行的应用程序访问 Amazon 资源。	ec2-sysadmin-*	Amazon EC2	授予 S3 存储桶访问权限的角色的示例策略，如《Amazon EC2 用户指南》中所示；可根据需要自定义
允许 Lambda 读取 DynamoDB Streams 并写入 CloudWatch Logs	lambda-sysadmin-*	Amazon Lambda	AWSLambdaDynamoDBExecutionRole

“仅查看”用户任务函数

Amazon 托管策略名称：ViewOnlyAccess

使用案例：此用户可以查看账户中服务的 Amazon 资源和基本元数据的列表。用户不得读取超出资源配额和列表信息的资源内容和元数据。

策略更新：Amazon 维护和更新此策略。有关此策略的更改历史记录，请在 IAM 控制台中查看策略，然后选择 Policy versions（策略版本）选项卡。有关任务函数策略更新的更多信息，请参阅 工作职能的 Amazon 托管策略。

策略描述：此策略授予针对 Amazon 服务的资源的 `List*`、`Describe*`、`Get*`、`View*` 和 `Lookup*` 访问权限。要查看此策略包含针对每种服务的哪些操作，请参阅 ViewOnlyAccess。有关托管式策略的更多信息，请参阅《Amazon 托管式策略参考指南》中的 ViewOnlyAccess。

工作职能的 Amazon 托管策略

这些策略均由 Amazon 维护并随时保持更新，以便在 Amazon 添加新服务和新功能时，包括对它们的支持。这些策略不能由客户修改。您可以复制策略并对副本进行修改，但在 Amazon 推出新的服务和 API 操作时，不会自动更新该副本。

对于任务函数策略，您可以在 IAM 控制台中查看版本历史记录以及每次更新的时间和日期。为此，请使用此页上的链接查看策略详细信息。然后选择 Policy versions（策略版本）选项卡以查看版本。此页显示策略的最新 25 个版本。要查看策略的所有版本，请调用 get-policy-version Amazon CLI 命令或 GetPolicyVersion API 操作。

> **Note**
>
> 您最多可以有五个版本的客户托管策略，但 Amazon 会保留 Amazon 托管策略的完整版本历史记录。

创建角色并附加策略（控制台）

上文列出的一些策略授予了通过角色配置 Amazon 服务的能力，以便这些服务能够代表您执行操作。工作职能策略或者指定您必须使用的确切角色名称，或者至少包括指定可用名称的开头部分的前缀。要创建这些角色之一，请执行以下程序中的步骤。

创建用于 Amazon Web Services 服务 的角色（IAM 控制台）

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
3. 对于 Trusted entity type（可信实体类型），选择 Amazon Web Services 服务。
4. 对于服务或使用案例，请选择服务，然后选择使用案例。用例由服务定义以包含服务要求的信任策略。
5. 选择下一步。
6. 对于权限策略，选项取决于您选择的使用案例：

- 如果服务定义了角色的权限，则您无法选择权限策略。
- 从一组有限的权限策略中进行选择。
- 从所有权限策略中进行选择。
- 不选择任何权限策略，创建角色后创建策略，然后将这些策略附加到该角色。

7. （可选）设置权限边界。这是一项高级功能，可用于服务角色，但不可用于服务相关角色。

   a. 打开设置权限边界部分，然后选择使用权限边界控制最大角色权限。

      IAM 包括您的账户中的 Amazon 托管式策略和客户管理型策略的列表。

   b. 选择要用于权限边界的策略。

8. 选择下一步。
9. 对于角色名称，选项取决于服务：

   - 如果服务定义角色名称，则您无法编辑角色名称。
   - 如果服务定义角色名称的前缀，您可以输入可选的后缀。
   - 如果服务未定义角色名称，您可以为该角色命名。

   > **Important**
   >
   > 命名角色时，请注意以下事项：
   >
   > - 角色名称在您的 Amazon Web Services 账户 中必须是唯一的，且不能因大小写而变得唯一。
   >
   >   例如，不要同时创建名为 **PRODROLE** 和 **prodrole** 的角色。当角色名称在策略中使用或者作为 ARN 的一部分时，角色名称区分大小写，但是当角色名称在控制台中向客户显示时（例如，在登录期间），角色名称不区分大小写。
   > - 创建角色后，您无法编辑该角色的名称，因为其他实体可能会引用该角色。

10. （可选）对于描述，输入角色的描述。
11. （可选）要编辑角色的使用案例和权限，请在步骤 1：选择可信实体或步骤 2：添加权限部分中选择编辑。
12. （可选）为了帮助识别、组织或搜索角色，请以键值对形式添加标签。有关在 IAM 中使用标签的更多信息，请参阅《IAM 用户指南》中的 Amazon Identity and Access Management 资源的标签。
13. 检查该角色，然后选择创建角色。

示例 1：将用户配置为数据库管理员（控制台）

此示例显示将 IAM 用户 Alice 配置为 Database Administrator（数据库管理员）需要执行的步骤。您需要使用此部分中的表中第一行中的信息，并允许该用户启用 Amazon RDS 监控。您将 DatabaseAdministrator 策略附加给 Alice 的 IAM 用户，以便他们可以管理 Amazon 数据库服务。该策略还允许 Alice 将名为 `rds-monitoring-role` 的角色传递给 Amazon RDS 服务，从而允许该服务代表他们监控 Amazon RDS 数据库。

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 选择策略，在搜索框中键入 **`database`**，然后按 Enter。
3. 选择 DatabaseAdministrator 策略对应的单选按钮，再选择操作，然后选择附加。
4. 在实体列表中选择 Alice，然后选择附加策略。Alice 现在可以管理 Amazon 数据库了。但是，要允许 Alice 监控这些数据库，您必须配置服务角色。
5. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
6. 请选择 Amazon Service（亚马逊云科技服务）角色类型，然后选择 Amazon RDS。
7. 请选择用于增强监控的 Amazon RDS 角色使用案例。
8. Amazon RDS 将为您的角色定义权限。选择下一步：审核以继续。
9. 角色名称必须是 Alice 当前拥有的 DatabaseAdministrator 策略中指定的一个。其中一个是 **`rds-monitoring-role`**。针对 Role name（角色名称）输入该信息。
10. （可选）对于角色描述，输入新角色的描述。
11. 在检查详细信息后，选择 Create role。
12. 现在，Alice 即可在 Amazon RDS 控制台的 Monitoring（监控）部分中启用 RDS Enhanced Monitoring（RDS 增强监控）。例如，他们可以在创建数据库实例、创建只读副本或修改数据库实例时执行此操作。当他们将 Enable Enhanced Monitoring（启用增强监控）设置为 Yes（是）时，他们必须输入他们在 Monitoring Role（监控角色）框中创建的角色名称（rds-monitoring-role）。

示例 2：将用户配置为网络管理员（控制台）

此示例显示将 IAM 用户 Jorge 配置为 Network Administrator（网络管理员）需要执行的步骤。它使用该部分的表中的信息来允许 Jorge 监控往来于 VPC 的 IP 流量。它还允许 Jorge 将该信息记录在 CloudWatch Logs 日志中。您将 NetworkAdministrator 策略附加给 Jorge 的 IAM 用户，以便他们可以配置 Amazon 网络资源。该策略还允许 Jorge 在您创建流日志时，将名称以 `flow-logs*` 开头的角色

传递给 Amazon EC2。此场景与示例 1 不同，这里没有预定义的服务角色类型，因此您必须执行几个不同的步骤。

1. 登录 Amazon Web Services 管理控制台，然后通过以下网址打开 IAM 控制台：https://console.aws.amazon.com/iam/。
2. 在导航窗格中，选择策略，在搜索框中输入 **network**，然后按 Enter。
3. 选中 NetworkAdministrator 策略旁的单选按钮，然后依次选择操作和附加。
4. 在用户列表中选择 Jorge 旁边的复选框，然后选择附加策略。Jorge 现在可以管理 Amazon 网络资源了。但是，为了监控 VPC 中的 IP 流量，您必须配置服务角色。
5. 由于您需要创建的服务角色没有预定义的托管策略，您必须先创建一个。在导航窗格中，选择策略，然后选择创建策略。
6. 在策略编辑器部分，选择 JSON 选项，然后复制以下 JSON 策略文档中的文本。将该文本粘贴到 JSON 文本框中。

   JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Action": [
        "logs:CreateLogGroup",
        "logs:CreateLogStream",
        "logs:PutLogEvents",
        "logs:DescribeLogGroups",
        "logs:DescribeLogStreams"
      ],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

7. 解决策略验证过程中生成的任何安全警告、错误或常规警告，然后选择下一步。

> **Note**
>
> 您可以随时在可视化和 JSON 编辑器选项卡之间切换。不过，如果您进行更改或在可视化编辑器中选择下一步，IAM 可能会调整您的策略结构以针对可视化编辑器进行优化。有关更多信息，请参阅 调整策略结构。

8. 在查看并创建页面上，键入 **`vpc-flow-logs-policy-for-service-role`** 作为策略名称。查看此策略中定义的权限以查看您的策略授予的权限，然后选择创建策略以保存您的工作。

   将在托管策略列表中显示新策略，并已准备好附加该策略。
9. 在 IAM 控制台的导航窗格中，选择角色，然后选择创建角色。
10. 请选择 Amazon Service（亚马逊云科技服务）角色类型，然后选择 Amazon EC2。
11. 请选择 Amazon EC2 使用案例。
12. 在附加权限策略页面上，选择您之前创建的策略：vpc-flow-logs-policy-for-service-role，然后选择下一步：审核。
13. 必须使用 Jorge 当前拥有的 NetworkAdministrator 策略所允许的角色名称。允许以 `flow-logs-` 开头的任何名称。在此示例中，对于 Role name（角色名称），请输入 **`flow-logs-for-jorge`**。
14. （可选）对于角色描述，输入新角色的描述。
15. 在检查详细信息后，选择 Create role。
16. 现在可以配置本场景需要的信任策略。在角色页面上，选择 flow-logs-for-jorge 角色（名称，而不是复选框）。在新角色的详细信息页面上，选择 Trust relationships (信任关系) 选项卡，然后选择 Edit trust relationship (编辑信任关系)。
17. 将条目替换为 `ec2.amazonaws.com`，“Service”行改为读取，如下所示：

    ```
            "Service": "vpc-flow-logs.amazonaws.com"
    ```
18. Jorge 现在可以在 Amazon EC2 控制台中为 VPC 或子网创建流日志了。当创建流日志时，请指定 flow-logs-for-jorge 角色。该角色拥有创建日志并向其写入数据的权限。

Amazon 全局条件上下文密钥

当主体向 Amazon 发出请求时，Amazon 会将请求信息收集到请求上下文中。您可以使用 JSON 策略的 `Condition` 元素将请求上下文中的键与您在策略中指定的键值进行比较。请求信息由不同的来源提供，包括发出请求的主体、请求所针对的资源以及有关请求本身的元数据。

全局条件键可在所有 Amazon 服务中使用。虽然这些条件键可以在所有策略中使用，但该键并非在每个请求上下文中都可用。例如，只有当 Amazon 服务主体直接调用您的资源时，aws:SourceAccount 条件键才可用。要了解有关在请求上下文中包含全局键的情况的更多信息，请参阅每个键的可用性信息。

某些单独的服务创建自己的条件键，这些键在其他服务的请求上下文中可用。跨服务条件键是一种全局条件键，其包含与服务名称相匹配的前缀（例如 ec2: 或 lambda:），但可跨其他服务使用。

服务特定条件键是为与单独的 Amazon 服务一起使用而定义的。例如，Amazon S3 允许您使用 s3:VersionId 条件键编写策略，以限制对特定版本 Amazon S3 对象的访问。此条件键是该服务所独有的，这意味着其仅适用于对 Amazon S3 服务的请求。对于服务特定的条件键，请参阅 Amazon 服务的操作、资源和条件键，然后选择要查看其键的服务。

Note

如果您使用只在某些情况下可用的条件键，则可使用 IfExists 版本的条件运算符。如果请求上下文中缺少条件键，则策略将无法通过评估。例如，将以下条件块与 ...IfExists 运算符结合使用以在请求来自特定 IP 范围或特定 VPC 时进行匹配。如果请求上下文中未包含这两个键或其中之一，则条件仍将返回 true。仅当请求上下文中包含指定的键时，才检查值。有关在其他运算符不存在键时如何评估策略的更多信息，请参阅条件运算符。

```
"Condition": {
    "IpAddressIfExists": {"aws:SourceIp" : ["xxx"] },
    "StringEqualsIfExists" : {"aws:SourceVpc" : ["yyy"]}
}
```

Important

要将条件与具有多个键值的请求上下文进行比较，必须使用 ForAllValues 或者 ForAnyValue 集合运算符。仅将集合运算符用于多值条件键。不要将集合运算符用于单值条件键。有关更多信息，请参阅 多值上下文键的集合运算符。

主体的属性	角色会话的属性	网络的属性	资源的属性	请求的属性
aws:PrincipalArn	aws:AssumedRoot	aws:SourceIp	aws:ResourceAccount	aws:CalledVia

主体的属性	角色会话的属性	网络的属性	资源的属性	请求的属性
aws:PrincipalAccount aws:PrincipalOrgPaths aws:PrincipalOrgID aws:PrincipalTag/ *tag-key* aws:PrincipalIsAWSService aws:PrincipalServiceName aws:PrincipalServiceNamesList aws:PrincipalType aws:userid aws:username	aws:FederatedProvider aws:TokenIssueTime aws:MultiFactorAuthAge aws:MultiFactorAuthPresent aws:ChatbotSourceArn aws:Ec2InstanceSourceVpc aws:Ec2InstanceSourcePrivateIPv4 aws:SourceIdentity ec2:RoleDelivery ec2:SourceInstanceArn	aws:SourceVpc aws:SourceVpcArn aws:SourceVpce aws:VpceAccount aws:VpceOrgID aws:VpceOrgPaths aws:VpcSourceIp	aws:ResourceOrgID aws:ResourceOrgPaths aws:ResourceTag/ *tag-key*	aws:CalledViaFirst aws:CalledViaLast aws:CalledViaAWSMCP aws:ViaAWSService aws:ViaAWSMCPService aws:CurrentTime aws:EpochTime aws:referer aws:RequestedRegion aws:RequestTag/ *tag-key* aws:TagKeys aws:SecureTransport aws:SourceAccount

主体的属性	角色会话的属性	网络的属性	资源的属性	请求的属性
	glue:RoleAssumedBy glue:CredentialIssuingService codebuild:BuildArn codebuild:ProjectArn lambda:SourceFunctionArn ssm:SourceInstanceArn identitystore:UserId			aws:SourceArn aws:SourceOrgID aws:SourceOrgPaths aws:UserAgent aws:IsMcpServiceAction

敏感条件键

以下条件键被视为敏感信息。在这些条件键中使用通配符没有任何有效的使用场景，即使键值的子字符串带有通配符也是如此。这是因为通配符可以将条件键与任何值匹配，这可能会带来安全风险。

- aws:PrincipalAccount
- aws:PrincipalOrgID
- aws:ResourceAccount
- aws:ResourceOrgID

- aws:SourceAccount
- aws:SourceOrgID
- aws:SourceVpc
- aws:SourceVpce
- aws:VpceAccount
- aws:VpceOrgID

主体的属性

使用以下条件键可将有关发出请求的主体的详细信息与您在策略中指定的主体属性进行比较。有关可以发出请求的主体的列表，请参阅 如何指定主体。

aws:PrincipalArn

使用此键可将发出请求的主体的 Amazon 资源名称 (ARN) 与您在策略中指定的 ARN 进行比较。对于 IAM 角色，请求上下文将返回角色的 ARN，而不是已代入角色的用户的 ARN。

- Availability（可用性）- 此键包含在所有签名请求的请求上下文中。匿名请求不包括此键。您可以在此条件键中指定以下类型的主体：
  - IAM 角色
  - IAM 用户
  - Amazon STS 联合用户主体
  - Amazon Web Services 账户 根用户
- 数据类型 – ARN

  Amazon 建议在比较 ARN 时使用 ARN 运算符而不是字符串运算符。
- 值类型 - 单值
- 示例值下表显示了您可以在 aws:PrincipalArn 条件键中指定的为不同类型的主体返回的请求上下文值：
  - IAM 角色 – 请求上下文包含条件键 aws:PrincipalArn 的以下值。请勿将所担任角色会话 ARN 指定为此条件键的值。有关所担任角色会话主体的更多信息，请参阅 角色会话主体。

    ```
    arn:aws:iam::123456789012:role/role-name
    ```
  - IAM 用户 – 请求上下文包含条件键 aws:PrincipalArn 的以下值。

```
arn:aws:iam::123456789012:user/user-name
```

- Amazon STS 联合用户主体 – 请求上下文包含条件键 aws:PrincipalArn 的以下值。

```
arn:aws:sts::123456789012:federated-user/user-name
```

- Amazon Web Services 账户 根用户 – 请求上下文包含条件键 aws:PrincipalArn 的以下值。当您将根用户 ARN 指定为 aws:PrincipalArn 条件键，它仅限制 Amazon Web Services 账户 根用户的权限。这不同于在基于资源策略的主体元素中指定根用户 ARN，后者会将权限委托给 Amazon Web Services 账户。有关在基于资源策略的主体元素中指定根用户 ARN 的更多信息，请参阅 Amazon Web Services 账户 主体。

```
arn:aws:iam::123456789012:root
```

您可以将根用户 ARN 指定为 Amazon Organizations 服务控制策略（SCP）条件键 aws:PrincipalArn 的值。SCP 是一种组织策略，用于管理组织中的权限，仅影响组织中的成员账户。SCP 会限制成员账户中的 IAM 用户和角色的权限，包括成员账户的根用户。有关 SCP 对权限影响的更多信息，请参阅《Amazon Organizations 用户指南》中的SCP 对权限的影响。

aws:PrincipalAccount

使用此键可将请求主体所属的账户与您在策略中指定的账户标识符进行比较。对于匿名请求，请求上下文将返回 anonymous。

- Availability（可用性）– 此键包含在所有请求（包括匿名请求）的请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值

在以下示例中，除了拥有账号 123456789012 的主体外，访问会被拒绝。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyAccessFromPrincipalNotInSpecificAccount",
```

```
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:us-east-1:111122223333:resource"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:PrincipalAccount": [
            "123456789012"
          ]
        }
      }
    }
  ]
}
```

aws:PrincipalOrgPaths

使用此键可将发出请求的主体的 Amazon Organizations 路径与策略中的路径进行比较。该主体可以是 IAM 用户、IAM 角色、Amazon STS 联合用户主体或 Amazon Web Services 账户根用户。在策略中，此条件键可确保请求者是 Amazon Organizations 中指定组织根或组织单元 (OU) 的账户成员。Amazon Organizations 路径是 Amazon Organizations 实体结构的文本表示形式。有关使用和了解路径的更多信息，请参阅了解 Amazon Organizations 实体路径。

- Availability（可用性）- 仅在主体是企业成员时，才将此键包含在请求上下文中。匿名请求不包括此键。
- 数据类型 – 字符串（列表）
- 值类型— 多值

> **Note**
>
> 组织 ID 是全局唯一的，但 OU ID 和根目录 ID 仅在组织内是唯一的。这意味着没有两个组织具有相同的组织 ID。但是，另一个组织可能具有与您的组织相同的 OU 或根目录 ID。我们建议您在指定 OU 或根目录时始终包含组织 ID。

例如，以下条件返回账户中主体的 `true`，该主体直接附加到 `ou-ab12-22222222` OU 但不在其子 OU 中。

```
"Condition" : { "ForAnyValue:StringEquals" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

以下条件为账户中的主体返回 `true`，该主体直接附加到 OU 或其任何子 OU。包含通配符时，您必须使用 `StringLike` 条件运算符。

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
*"]
}}
```

以下条件为账户中的主体返回 `true`，该主体直接附加到任意子 OU，但不会直接附加至父 OU。以前的条件适用于 OU 或任何子 OU。以下条件仅适用于子 OU（以及这些子 OU 的任何子 OU）。

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/
ou-*"]
}}
```

以下条件允许访问 `o-a1b2c3d4e5` 组织中的各个主体，而无论其父 OU 如何。

```
"Condition" : { "ForAnyValue:StringLike" : {
     "aws:PrincipalOrgPaths":["o-a1b2c3d4e5/*"]
}}
```

`aws:PrincipalOrgPaths` 是一个多值条件键。多值键在请求上下文中可以有多个值。在您为 `ForAnyValue` 条件运算符使用多个值时，主体的路径必须与策略中列出的其中一个路径匹配。有关多值条件键的更多信息，请参阅[多值上下文键的集合运算符]。

```
    "Condition": {
        "ForAnyValue:StringLike": {
            "aws:PrincipalOrgPaths": [
                "o-a1b2c3d4e5/r-ab12/ou-ab12-33333333/*",
                "o-a1b2c3d4e5/r-ab12/ou-ab12-22222222/*"
            ]
        }
    }
```

aws:PrincipalOrgID

使用此键可将请求主体所属的 Amazon Organizations 中组织的标识符与策略中指定的标识符进行比较。

- Availability（可用性）- 仅在主体是企业成员时，才将此键包含在请求上下文中。匿名请求不包括此键。
- 数据类型 – 字符串
- 值类型— 单值

此全局键提供了列出组织中的所有 Amazon 账户的所有账户 ID 的替代方法。您可以使用此条件键来简化在基于资源的策略中指定 `Principal` 元素的过程。您可以在条件元素中指定组织 ID。当您添加和删除账户时，包含 `aws:PrincipalOrgID` 键的策略将自动包括正确的账户，并且不需要手动更新。

例如，以下 Amazon S3 存储桶策略允许 `o-xxxxxxxxxxx` 企业中的任何账户成员将对象添加到 `amzn-s3-demo-bucket` 存储桶中。

JSON

```
{
 "Version":"2012-10-17",
 "Statement": {
   "Sid": "AllowPutObject",
   "Effect": "Allow",
   "Principal": "*",
   "Action": "s3:PutObject",
   "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
   "Condition": {"StringEquals":
     {"aws:PrincipalOrgID":"o-xxxxxxxxxxx"}
   }
 }
}
```

> **Note**
>
> 此全局条件也适用于 Amazon 企业的管理账户。此策略防止指定组织之外的所有主体访问 Amazon S3 存储桶。这包括与您的内部资源交互的任何 Amazon 服务，如将日志数据发送到

Amazon S3 存储桶的 Amazon CloudTrail。了解如何安全地授予 Amazon 服务的访问权限，请参阅 aws:PrincipalIsAWSService。

有关 Amazon Organizations 的更多信息，请参阅《Amazon Organizations 用户指南》中的 什么是 Amazon Organizations？。

aws:PrincipalTag/tag-key

使用此键可将附加到发出请求的主体的标签与您在策略中指定的标签进行比较。如果已为主体附加多个标签，则请求上下文会为每个附加的标签键包含一个 `aws:PrincipalTag` 键。

- 可用性 - 仅在主体使用具有附加标签的 IAM 用户时，才将此键包含在请求上下文中。对于主体，如果使用附加了标签或会话标签的 IAM 角色，则应包括它。匿名请求不包括此键。
- 数据类型 – 字符串
- 值类型— 单值

您可采用键/值对的形式向用户或角色添加自定义属性。有关 IAM 标签的更多信息，请参阅 Amazon Identity and Access Management 资源的标签。您可以将 `aws:PrincipalTag` 用于针对 Amazon 主体的访问控制。

此示例说明如何创建基于角色的策略，以允许具有 **`department=hr`** 标签的用户管理 IAM 用户、组或角色。要使用此策略，请将示例策略中的*#######*替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "iam:Get*",
        "iam:List*",
        "iam:Generate*"
      ],
      "Resource": "*",
      "Condition": {
```

```
        "StringEquals": {
          "aws:PrincipalTag/department": "hr"
        }
      }
    }
  ]
}
```

aws:PrincipalIsAWSService

使用此密钥可检查对您资源的调用是否直接由 Amazon 服务主体提出。例如，Amazon CloudTrail 使用服务主体 `cloudtrail.amazonaws.com` 将日志写入 Amazon S3 存储桶。当服务使用服务主体对资源执行直接操作时，请求上下文密钥将设置为 true。如果服务使用 IAM 主体的凭证代表主体发出请求时，上下文密钥设置为 false。如果服务使用服务角色或服务相关角色代表主体进行调用，则请求也会被设为 false。

- 可用性 — 此密钥存在于所有使用 Amazon 凭证的签名 API 请求的请求上下文中。匿名请求不包括此键。
- 数据类型 – 布尔值
- 值类型— 单值

您可以使用此条件键限制对可信身份和预期网络位置的访问，同时安全地授予对 Amazon 服务的访问权限。

在以下 Amazon S3 存储桶策略示例中，对存储桶的访问受到限制，除非请求源自 `vpc-111bbb22` 或者来自服务主体，例如 CloudTrail。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "ExpectedNetworkServicePrincipal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/AmazonLogs/AccountNumber/*",
      "Condition": {
```

```
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22"
        },
        "BoolIfExists": {
          "aws:PrincipalIsAWSService": "false"
        }
      }
    }
  ]
}
```

aws:PrincipalServiceName

使用此密钥可以将策略中的服务主体名称与正在向您的资源发出请求的服务主体进行比较。您可以使用此密钥来检查此调用是否由特定的服务主体提出。当服务主体向您的资源发出直接请求时，aws:PrincipalServiceName 密钥将包含服务主体的名称。例如，Amazon CloudTrail 服务主体的名称是 cloudtrail.amazonaws.com。

- 可用性 — 当调用由 Amazon 服务主体提出时，此密钥将存在于请求中。此密钥不会在任何其他情况中出现，包括以下情况：
  - 如果服务使用服务角色或服务相关角色代表主体进行调用。
  - 如果服务使用 IAM 主体的凭证代表主体发出请求。
  - 如果调用是由 IAM 主体直接发出的。
  - 如果调用是由匿名请求者发出的。
- 数据类型 – 字符串
- 值类型— 单值

您可以使用此条件键限制对可信身份和预期网络位置的访问，同时安全地授予对 Amazon 服务的访问权限。

在以下 Amazon S3 存储桶策略示例中，对存储桶的访问受到限制，除非请求源自 vpc-111bbb22 或者来自服务主体，例如 CloudTrail。

JSON

```
{
  "Version":"2012-10-17",
```

```
  "Statement": [
    {
      "Sid": "ExpectedNetworkServicePrincipal",
      "Effect": "Deny",
      "Principal": "*",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/AmazonLogs/AccountNumber/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-111bbb22",
          "aws:PrincipalServiceName": "cloudtrail.amazonaws.com"
        }
      }
    }
  ]
}
```

aws:PrincipalServiceNamesList

此密钥提供了属于此服务的所有服务主体名称列表。这是一个高级条件键。您可以用它来限制服务仅从特定地区访问您的资源。某些服务可能会创建区域服务主体，以指示特定区域内服务的特定实例。您可以将对资源的访问限制为服务的特定实例。当服务主体向您的资源发出直接请求时，aws:PrincipalServiceNamesList 会包含与服务的区域实例关联的所有服务主体名称的无序列表。

- 可用性 — 当调用由 Amazon 服务主体提出时，此密钥将存在于请求中。此密钥不会在任何其他情况中出现，包括以下情况：
  - 如果服务使用服务角色或服务相关角色代表主体进行调用。
  - 如果服务使用 IAM 主体的凭证代表主体发出请求。
  - 如果调用是由 IAM 主体直接发出的。
  - 如果调用是由匿名请求者发出的。
- 数据类型 – 字符串（列表）
- 值类型— 多值

aws:PrincipalServiceNamesList 是一个多值条件键。多值键在请求上下文中可以有多个值。对于此键，您必须将 ForAnyValue 或者 ForAllValues 集合运算符与字符串条件运算符搭配使用。有关多值条件键的更多信息，请参阅多值上下文键的集合运算符。

aws:PrincipalType

使用此键可将发出请求的主体的类型与您在策略中指定的主体类型进行比较。有关更多信息，请参阅如何指定主体。有关 `principal` 密钥值的具体示例，请参阅 主体键值。

- Availability（可用性）– 此键包含在所有请求（包括匿名请求）的请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值

aws:userid

使用此键可将请求者的主体标识符与您在策略中指定的 ID 进行比较。对于 IAM 用户，请求上下文值是用户 ID。对于 IAM 角色，此值的格式可能有所不同。有关如何为不同的主体显示信息的详细信息，请参阅如何指定主体。有关 `principal` 密钥值的具体示例，请参阅 主体键值。

- Availability（可用性）– 此键包含在所有请求（包括匿名请求）的请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值

aws:username

使用此键可将请求者的用户名与您在策略中指定的用户名进行比较。有关如何为不同的主体显示信息的详细信息，请参阅如何指定主体。有关 `principal` 密钥值的具体示例，请参阅 主体键值。

- 可用性 - 此键始终包含在 IAM 用户的请求上下文中。匿名请求和使用 Amazon Web Services 账户根用户 或 IAM 角色发出的请求不包含此键。使用 IAM Identity Center 凭证发出的请求不会在上下文中包含此键。
- 数据类型 – 字符串
- 值类型 - 单值

角色会话的属性

使用以下条件键来比较生成会话时角色会话的属性。只有拥有角色会话或联合用户主体凭证的主体发出请求时，这些条件键才可用。这些条件键的值嵌入角色的会话令牌中。

角色是一种主体。您还可以使用 主体的属性 部分中的条件键在角色发出请求时评估角色的属性。

aws:AssumedRoot

使用此键可检查是否已使用 AssumeRoot 发出请求。AssumeRoot 返回特权根用户会话的短期凭证，您可以使用该凭证对组织中的成员账户执行特权操作。有关更多信息，请参阅 集中管理成员账户的根访问权限。

- 可用性 – 仅当主体使用来自 AssumeRoot 的凭证发出请求时，此键才包含在请求上下文中。
- 数据类型 – 布尔值
- 值类型 - 单值

在以下示例中，当用作服务控制策略时，拒绝在 Amazon Organizations 成员账户中使用根用户的长期凭证。该策略不会拒绝 AssumeRoot 会话采取 AssumeRoot 会话允许的操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement":[
       {
          "Effect":"Deny",
          "Action":"*",
          "Resource": "*",
          "Condition":{
             "ArnLike":{
                "aws:PrincipalArn":[
                   "arn:aws:iam::*:root"
                ]
             },
             "Null":{
                "aws:AssumedRoot":"true"
             }
          }
       }
    ]
 }
```

aws:FederatedProvider

使用此密钥将主体的发布身份提供者 (IdP) 与您在策略中指定的 IdP 进行比较。这意味着 IAM 角色使用 `AssumeRoleWithWebIdentity` Amazon STS 操作代入。使用生成的角色会话的临时证书发出请求时，请求上下文将标识对原始联合身份进行身份验证的 IdP。

- 可用性 – 此键存在于使用 OpenID Connect (OIDC) 提供程序代入的角色的角色会话中，以及当使用 OIDC 提供程序调用 `AssumeRoleWithWebIdentity` 时的角色信任策略中。
- 数据类型 – 字符串*
- 值类型 - 单值

* 数据类型取决于您的 IdP：

- 如果您使用的是内置 Amazon IdP，比如 Amazon Cognito，则密钥值将是一个字符串。密钥值可能如下所示：`cognito-identity.amazonaws.com`。
- 如果您使用的是没有内置于 Amazon 的 IdP 、比如 GitHub 或 Amazon EKS，则密钥值将为 ARN。密钥值可能如下所示：`arn:aws:iam::111122223333:oidc-provider/oidc.eks.region.amazonaws.com/id/OIDC_Provider_ID`。

有关外部 IdP 和 `AssumeRoleWithWebIdentity` 的更多信息，请参阅 常见场景。有关更多信息，请参阅 角色会话主体。

aws:TokenIssueTime

使用此键可将临时安全凭证的颁发日期和时间与您在策略中指定的日期和时间进行比较。

- Availability（可用性）- 仅在主体使用临时凭证发出请求时，才将此键包含在请求上下文中。使用访问密钥发出的 Amazon CLI、Amazon API 或 Amazon 开发工具包请求中未提供此键。
- 数据类型 – 日期
- 值类型— 单值

要了解哪些服务支持使用临时凭证，请参阅使用 IAM 的 Amazon 服务。

aws:MultiFactorAuthAge

使用此键可将自使用 MFA 向请求主体授权以来的秒数与您在策略中指定的数量进行比较。有关 MFA 的更多信息，请参阅 IAM 中的 Amazon 多重身份验证。

> **Important**
>
> 对于联合身份或使用访问密钥签署 Amazon CLI、Amazon API 或 Amazon SDK 请求而发出的请求，此条件键不存在。要了解有关使用临时安全凭证为 API 操作添加 MFA 保护的更多信息，请参阅 使用 MFA 保护 API 访问。
> 要检查是否使用 MFA 来验证 IAM 联合身份，您可以将身份提供者的身份验证方法作为会话标签传递给 Amazon。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。要对 IAM Identity Center 身份强制执行 MFA，您可以启用访问控制属性，以使用身份提供者的身份验证方法将 SAML 断言声明传递给 IAM Identity Center。

- 可用性 – 仅在主体使用临时安全凭证发出请求时，才将此键包含在请求上下文中。带 MFA 条件的策略可附加到：
  - 对于 IAM 用户或组：
  - Amazon S3 存储桶、Amazon SQS 队列或 Amazon SNS 主题等资源
  - 可由用户担任的 IAM 角色的信任策略
- 数据类型 – 数字
- 值类型 - 单值

aws:MultiFactorAuthPresent

使用此键可检查是否已使用多重身份验证（MFA）来验证发出请求的临时安全凭证。

> **Important**
>
> 对于联合身份或使用访问密钥签署 Amazon CLI、Amazon API 或 Amazon SDK 请求而发出的请求，此条件键不存在。要了解有关使用临时安全凭证为 API 操作添加 MFA 保护的更多信息，请参阅 使用 MFA 保护 API 访问。
> 要检查是否使用 MFA 来验证 IAM 联合身份，您可以将身份提供者的身份验证方法作为会话标签传递给 Amazon。有关更多信息，请参阅 在 Amazon STS 中传递会话标签。要对 IAM Identity Center 身份强制执行 MFA，您可以启用访问控制属性，以使用身份提供者的身份验证方法将 SAML 断言声明传递给 IAM Identity Center。

- Availability（可用性）- 仅在主体使用临时凭证发出请求时，才将此键包含在请求上下文中。带 MFA 条件的策略可附加到：

  - 对于 IAM 用户或组：
  - Amazon S3 存储桶、Amazon SQS 队列或 Amazon SNS 主题等资源
  - 可由用户担任的 IAM 角色的信任策略
- 数据类型 – 布尔值
- 值类型 - 单值

临时凭证用于验证具有来自 AssumeRole 或 GetSessionToken 的临时令牌的 IAM 角色和 IAM 用户，以及 Amazon Web Services 管理控制台 用户。

IAM 用户访问密钥是长期凭证，但在某些情况下，Amazon 会代表 IAM 用户创建临时凭证以执行操作。在这些情况下，`aws:MultiFactorAuthPresent` 密钥存在于请求中，并设置为值 `false`。有两种常见情况可能会发生这种情形：

- Amazon Web Services 管理控制台中的 IAM 用户在不知情的情况下使用了临时凭证。用户使用其作为长期凭证的用户名和密码登录控制台。但在后台，控制台代表用户生成临时凭证。
- 如果 IAM 用户对某个 Amazon 服务进行调用，则该服务将重新使用该用户的凭证向其他服务发出另一个请求。例如，当调用 Athena 以访问 Amazon S3 存储桶或使用 Amazon CloudFormation 创建 Amazon EC2 实例时。对于后续请求，Amazon 使用临时凭证。

要了解哪些服务支持使用临时凭证，请参阅使用 IAM 的 Amazon 服务。

在使用长期凭证（例如，用户访问密钥对）调用 API 或 CLI 命令时，不提供 `aws:MultiFactorAuthPresent` 键。因此我们建议，当您检查此键时使用 `...IfExists` 版本的条件运算符。

必须理解，以下 `Condition` 元素不是 检查请求是否使用 MFA 进行身份验证的可靠方式。

```
#####   WARNING: NOT RECOMMENDED   #####
"Effect" : "Deny",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "false" } }
```

`Deny` 效果、`Bool` 元素和 `false` 值的这一组合可拒绝可以使用但未使用 MFA 进行身份验证的请求。这仅适用于支持使用 MFA 的临时凭证。此语句不会拒绝对使用长期凭证发出的请求或使用 MFA 进行身份验证的请求的访问权限。使用此示例时务必要谨慎，因为其逻辑复杂，并且未对是否实际使用 MFA 身份验证进行测试。

此外，请勿使用 Deny 效果、Null 元素和 true 的组合，因为该组合行为方式相同，但逻辑更加复杂。

建议的组合

我们建议您使用 BoolIfExists 运算符检查是否使用 MFA 对请求进行身份验证。

```
"Effect" : "Deny",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Deny、BoolIfExists 和 false 的组合将拒绝未使用 MFA 进行身份验证的请求。具体来说，它拒绝来自不包含 MFA 的临时凭证的请求。它还拒绝使用长期凭证发出的请求，例如使用访问密钥执行的 Amazon CLI 或 Amazon API 操作。*IfExists 运算符用于检查 aws:MultiFactorAuthPresent 键是否存在以及它是否可以存在（由其存在性指示）。当您要拒绝未使用 MFA 进行身份验证的任何请求时，可使用此运算符。这是更安全的，但可能会中断使用访问密钥访问 Amazon CLI 或 Amazon API 的任何代码或脚本。

替代组合

您还可以使用 BoolIfExists 运算符以允许通过 MFA 进行身份验证的请求以及使用长期凭证发出的 Amazon CLI 或 Amazon API 请求。

```
"Effect" : "Allow",
"Condition" : { "BoolIfExists" : { "aws:MultiFactorAuthPresent" : "true" } }
```

此条件将匹配键存在或键不存在的情况。Allow、BoolIfExists 和 true 的组合允许已使用 MFA 进行身份验证的请求或无法使用 MFA 进行身份验证的请求。这意味着，在请求者使用其长期访问密钥时，允许执行 Amazon CLI、Amazon API 和 Amazon 开发工具包操作。该组合不允许来自临时凭证的请求，这些凭证可能包含 MFA，但未包含 MFA。

在使用 IAM 控制台可视化编辑器创建策略并选择需要 MFA 时，将应用该组合。该设置要求使用 MFA 进行控制台访问，但允许在没有 MFA 的情况下进行编程访问。

或者，您也可以使用 Bool 运算符，以便仅在使用 MFA 进行身份验证时允许编程和控制台请求。

```
"Effect" : "Allow",
"Condition" : { "Bool" : { "aws:MultiFactorAuthPresent" : "true" } }
```

Allow、Bool 和 true 的组合仅允许已进行 MFA 身份验证的请求。这仅适用于支持使用 MFA 的临时凭证。该语句不允许访问使用长期访问密钥发出的请求，也不允许访问在没有 MFA 的情况下使用临时凭证发出的请求。

请勿使用策略进行类似以下的构建，以检查 MFA 密钥是否存在：

```
#####   WARNING: USE WITH CAUTION   #####

"Effect" : "Allow",
"Condition" : { "Null" : { "aws:MultiFactorAuthPresent" : "false" } }
```

Allow 效果、Null 元素和 false 值的组合仅允许可以使用 MFA 进行身份验证的请求，而不考虑该请求是否已实际进行身份验证。这组合允许使用临时凭证发出的所有请求，而对长期凭证拒绝访问权限。使用此示例时务必要谨慎，因为它未对是否实际使用 MFA 身份验证进行测试。

aws:ChatbotSourceArn

使用此键，可将主体设置的来源聊天配置 ARN 跟您在与频道配置关联的 IAM 角色策略中指定的聊天配置 ARN 进行比较。您可以根据聊天应用程序中的 Amazon Q 开发者版发起的代入角色会话对请求进行授权。

- 可用性 – 每次代入角色会话时，此键都将包含在聊天应用程序中的 Amazon Q 开发者版服务发起的请求上下文中。键值是聊天配置 ARN，例如从聊天频道运行 Amazon CLI 命令时。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws::chatbot::123456789021:chat-configuration/slack-channel/private_channel

以下策略拒绝 Amazon S3 将所有来自 Slack 频道的请求放入指定存储桶。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleS3Deny",
            "Effect": "Deny",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
                "ArnLike": {
```

```
                    "aws:ChatbotSourceArn": "arn:aws:chatbot::*:chat-
configuration/slack-channel/*"
                }
            }
        }
    ]
}
```

aws:Ec2InstanceSourceVpc

此键标识将 Amazon EC2 IAM 角色凭证传递到的 VPC。您可以在具有 aws:SourceVPC 全局键的策略中使用此键，检查从中发出调用的 VPC (aws:SourceVPC) 是否与将凭证传送到的 VPC (aws:Ec2InstanceSourceVpc) 相匹配。

- 可用性 – 仅在请求者使用 Amazon EC2 角色凭证签署请求时，才将此键包含在请求上下文中。它可以用于 IAM 策略、服务控制策略、VPC 端点策略和资源策略。
- 数据类型 – 字符串
- 值类型 - 单值

此键可以与 VPC 标识符值一起使用，但在用作与 aws:SourceVpc 上下文键组合的变量时最有用。只有当请求者使用 VPC 端点发出请求时，才将 aws:SourceVpc 上下文键包含在请求上下文中。搭配使用 aws:Ec2InstanceSourceVpc 和 aws:SourceVpc 时，您能够更广泛地使用 aws:Ec2InstanceSourceVpc，因为它可以比较通常一起变化的值。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "RequireSameVPC",
      "Effect": "Deny",
      "Action": "*",
      "Resource": "*",
      "Condition": {
        "StringNotEquals": {
            "aws:SourceVpc": "${aws:Ec2InstanceSourceVpc}"
        },
        "Null": {
```

```
          "ec2:SourceInstanceARN": "false"
        },
        "BoolIfExists": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

在上面的示例中，如果 aws:SourceVpc 值不等于 aws:Ec2InstanceSourceVpc 值，则访问将被拒绝。通过测试 ec2:SourceInstanceARN 条件键是否存在，该策略声明仅限于用作 Amazon EC2 实例角色的角色。

该策略使用 aws:ViaAWSService，在代表您的 Amazon EC2 实例角色发出请求时允许 Amazon 对请求进行授权。例如，当您从 Amazon EC2 实例向加密的 Amazon S3 存储桶发出请求时，Amazon S3 会代表您调用 Amazon KMS。向 Amazon KMS 发出请求时，某些键不存在。

aws:Ec2InstanceSourcePrivateIPv4

此键标识 Amazon EC2 IAM 角色凭证传送到的主弹性网络接口的私有 IPv4 地址。您必须将此条件键与其配套键 aws:Ec2InstanceSourceVpc 一起使用，以确保您拥有 VPC ID 和源私有 IP 的全局唯一组合。将此键与 aws:Ec2InstanceSourceVpc 搭配使用可确保请求是从凭证传送到的同一私有 IP 地址发出的。

- 可用性 – 仅在请求者使用 Amazon EC2 角色凭证签署请求时，才将此键包含在请求上下文中。它可以用于 IAM 策略、服务控制策略、VPC 端点策略和资源策略。
- 数据类型 – IP 地址
- 值类型 - 单值

> **Important**
>
> 此键不应在 Allow 语句中单独使用。顾名思义，私有 IP 地址不是全局唯一的。每次使用 aws:Ec2InstanceSourcePrivateIPv4 键来指定可以从哪个 VPC 使用 Amazon EC2 实例凭证时，都应使用 aws:Ec2InstanceSourceVpc 键。

以下示例是服务控制策略（SCP），除非请求通过与角色凭证位于同一 VPC 中的 VPC 端点到达，否则该策略将拒绝访问所有资源。在此示例中，aws:Ec2InstanceSourcePrivateIPv4 根据源 IP 将凭证源限制为特定实例。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourceVpc": "${aws:SourceVpc}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
                    "aws:ViaAWSService": "false"
                }
            }
        },
        {
            "Effect": "Deny",
            "Action":  "*",
            "Resource": "*",
            "Condition": {
                "StringNotEquals": {
                    "aws:Ec2InstanceSourcePrivateIPv4": "${aws:VpcSourceIp}"
                },
                "Null": {
                    "ec2:SourceInstanceARN": "false"
                },
                "BoolIfExists": {
                    "aws:ViaAWSService": "false"
                }
            }
        }
    ]
```

```
}
```

aws:SourceIdentity

使用此密钥可将主体设置的源身份与您在策略中指定的源身份进行比较。

- 可用性 — 在使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS `AssumeRole`API 操作担任角色时设置好源身份后，此密钥将会被包含在请求中。
- 数据类型 – 字符串
- 值类型— 单值

您可以在策略中使用此密钥来允许 Amazon 中的主体操作，这些主体在担任角色时设置了源身份。角色指定源身份的活动显示在 Amazon CloudTrail。这使管理员能够更轻松地确定什么角色在 Amazon 中执行了操作。

与 `sts:RoleSessionName` 不同，在设置源身份后，无法更改该值。它存在于角色执行的所有操作的请求上下文中。当您使用会话凭证担任另一个角色时，该值将保留到后续角色会话中。从一个角色代入另一个角色的过程称为角色链。

当主体使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS `AssumeRole` API 操作担任角色并初次设置源身份时，`sts:SourceIdentity` 密钥将会出现在请求中。`aws:SourceIdentity`键存在于对具有源身份设置的角色会话执行的任何操作请求中。

以下账户 `111122223333` 中适用于 `CriticalRole` 的角色信任策略包含一个条件适用于 `aws:SourceIdentity` 的条件，可防止未将源身份设置为 Saanvi 或 Diego 主体担任该角色。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AssumeRoleIfSourceIdentity",
            "Effect": "Allow",
            "Principal": {"AWS": "arn:aws:iam::123456789012:role/CriticalRole"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
```

```
            ],
            "Condition": {
                "StringLike": {
                    "aws:SourceIdentity": ["Saanvi","Diego"]
                }
            }
        }
    ]
}
```

要了解有关使用源身份信息的更多信息，请参阅 监控和控制使用所担任角色执行的操作。

ec2:RoleDelivery

使用此键可将签名请求中的实例元数据服务版本与 Amazon EC2 的 IAM 角色凭证进行比较。实例元数据服务根据以下条件区分 IMDSv1 和 IMDSv2 请求：对于任何给定请求，PUT 或 GET 标头（对于 IMDSv2 是唯一的）在该请求中是否存在。

- 可用性 – 仅在 Amazon EC2 实例创建角色会话时，才将此键包含在请求上下文中。
- 数据类型 – 数字
- 值类型 - 单值
- 示例值 – 1.0, 2.0

您可以在每个实例上配置实例元数据服务（IMDS），以确保本地代码或用户必须使用 IMDSv2。在指定必须使用 IMDSv2 时，IMDSv1 不再起作用。

- 实例元数据服务版本 1（IMDSv1）– 一种请求/响应方法
- 实例元数据服务版本 2 (IMDSv2)：一种面向会话的方法

有关如何配置实例以使用 IMDSv2 的信息，请参阅配置实例元数据选项。

在以下示例中，如果请求上下文中的 ec2:RoleDelivery 值为 1.0（IMDSv1），则访问将被拒绝。此策略语句可广泛应用，因为如果请求未由 Amazon EC2 角色证书签名，则其为无效。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [
              {
           "Sid": "RequireAllEc2RolesToUseV2",
           "Effect": "Deny",
           "Action": "*",
           "Resource": "*",
           "Condition": {
               "NumericLessThan": {
                   "ec2:RoleDelivery": "2.0"
               }
           }
       }
    ]
}
```

有关更多信息，请参阅使用实例元数据的示例策略。

ec2:SourceInstanceArn

使用此键可以比较生成角色会话的实例的 ARN。

- 可用性 – 仅在 Amazon EC2 实例创建角色会话时，才将此键包含在请求上下文中。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws:ec2:us-west-2:111111111111:instance/instance-id

对于策略示例，请参阅允许特定实例查看其他 Amazon 服务中的资源。

glue:RoleAssumedBy

Amazon Glue 服务为每个 Amazon API 请求设置此条件键，其中 Amazon Glue 使用服务角色代表客户发出请求（不是通过作业或开发人员端点，而是直接通过 Amazon Glue 服务发出请求）。使用此键验证对 Amazon 资源的调用是否来自 Amazon Glue 服务。

- 可用性 – 当 Amazon Glue 代表客户使用服务角色发出请求时，此键包含在请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值
- 示例值 – 此键始终设置为 `glue.amazonaws.com`。

以下示例添加了一个允许 Amazon Glue 服务从 Amazon S3 存储桶获取对象的条件。

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:RoleAssumedBy": "glue.amazonaws.com"
        }
    }
}
```

glue:CredentialIssuingService

Amazon Glue 服务使用来自作业或开发人员端点的服务角色为每个 Amazon API 请求设置此键。使用此键验证对 Amazon 资源的调用是来自 Amazon Glue 作业还是开发人员端点。

- 可用性 – 当 Amazon Glue 发出来自作业或开发人员端点的请求时，此键包含在请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值
- 示例值 – 此键始终设置为 `glue.amazonaws.com`。

以下示例添加一个附加到 Amazon Glue 作业使用的 IAM 角色的条件。这确保了可以基于角色会话是否用于 Amazon Glue 任务运行时环境而允许/拒绝某些操作。

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "StringEquals": {
            "glue:CredentialIssuingService": "glue.amazonaws.com"
        }
    }
}
```

codebuild:BuildArn

此键标识将 IAM 角色凭证传递到的 CodeBuild 构建 ARN。使用此键可验证对 Amazon 资源的调用是否来自特定 CodeBuild 构建。

> **Note**
>
> 事先不知道 `codebuild:BuildArn` 的完整值，因为其中包含动态生成的构建 ID。

- 可用性 – 仅在由 CodeBuild 代入的角色发出请求时，将此键包含在请求上下文中。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws:codebuild:us-east-1:123456789012:build/MyBuildProject:12345678-1234-1234-1234-123456789012

以下示例允许特定 CodeBuild 构建让 `s3:GetObject` 访问指定的存储桶。

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "ArnLike": {
            "codebuild:BuildArn": "arn:aws:codebuild:us-east-1:123456789012:build/
MyBuildProject:*"
        }
    }
}
```

codebuild:ProjectArn

此键标识将 IAM 角色凭证传递到 CodeBuild 构建的 CodeBuild 项目 ARN。使用此键可验证对 Amazon 资源的调用是否来自特定 CodeBuild 项目。

- 可用性 – 仅在由 CodeBuild 代入的角色发出请求时，将此键包含在请求上下文中。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws:codebuild:us-east-1:123456789012:project/MyBuildProject

以下示例允许来自特定 CodeBuild 项目的任何构建让 `s3:GetObject` 访问指定的存储桶。

```
{
    "Effect": "Allow",
    "Action": "s3:GetObject",
    "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
    "Condition": {
        "ArnEquals": {
            "codebuild:ProjectArn": "arn:aws:codebuild:us-east-1:123456789012:project/
MyBuildProject"
        }
    }
}
```

lambda:SourceFunctionArn

使用此键标识将 IAM 角色凭证传递到的 Lambda 函数 ARN。Lambda 服务会为来自函数执行环境的每个 Amazon API 请求设置此键。使用此键可验证对 Amazon 资源的调用是否来自特定 Lambda 函数的代码。Lambda 还为来自执行环境外部的一些请求设置此键，例如将日志写入 CloudWatch 并将跟踪发送到 X-Ray。

- 可用性 – 每次调用 Lambda 函数代码时，此键都将包含在请求上下文中。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws:lambda:us-east-1:123456789012:function:TestFunction

以下示例允许一个特定的 Lambda 函数让 `s3:PutObject` 访问指定的存储桶。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "ExampleSourceFunctionArn",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket/*",
            "Condition": {
```

```
                "ArnEquals": {
                    "lambda:SourceFunctionArn": "arn:aws:lambda:us-
east-1:123456789012:function:source_lambda"
                }
            }
        }
    ]
}
```

有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的使用 Lambda 执行环境凭证。

ssm:SourceInstanceArn

使用此键标识将 IAM 角色凭证传递到的 Amazon Systems Manager 托管实例 ARN。请求来自使用与 Amazon EC2 实例配置文件关联的 IAM 角色的托管实例时，此条件键不存在。

- 可用性 – 每次将角色凭证传递到 Amazon Systems Manager 托管实例时，此键都将包含在请求上下文中。
- 数据类型 – ARN
- 值类型 - 单值
- 示例值 – arn:aws:ec2:us-west-2:111111111111:instance/instance-id

identitystore:UserId

使用此键可将签名请求中的 IAM Identy Center 员工身份与策略中指定的身份进行比较。

- 可用性 – 当请求的调用方是 IAM Identity Center 中的用户时，将包含此键。
- 数据类型 – 字符串
- 值类型 - 单值
- 示例值 – 94482488-3041-7026-18f3-be45837cd0e4

您可以使用 Amazon CLI、Amazon API 或 Amazon 软件开发工具包向 GetUserId API 发出请求，以在 IAM Identity Center 查找用户的用户 ID。

网络的属性

使用以下条件键可将有关请求源自或传递网络的详细信息与您在策略中指定的网络属性进行比较。

aws:SourceIp

使用此键可将请求者的 IP 地址与您在策略中指定的 IP 地址进行比较。aws:SourceIp 条件键只能用于公有 IP 地址范围。

- 可用性 – 此键将包含在请求上下文中，但请求者使用 VPC 端点发出请求时除外。
- 数据类型 – IP 地址
- 值类型— 单值

可在策略中使用 aws:SourceIp 条件键以仅允许主体从指定的 IP 范围发出请求。

> **Note**
>
> aws:SourceIp 同时支持 IPv4 和 IPv6 地址或 IP 地址范围。有关支持 IPv6 的 Amazon Web Services 服务 列表，请参阅《Amazon VPC 用户指南》中支持 IPv6 的 Amazon Web Services 服务。

例如，您可以将以下基于身份的策略附加到 IAM 角色。如果用户从指定的 IPv4 地址范围进行调用，此策略允许用户将对象放入 amzn-s3-demo-bucket3 Amazon S3 存储桶。此策略还允许使用 转发访问会话 的 Amazon 服务代表您执行此操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": "203.0.113.0/24"
                }
            }
        }
    ]
```

```
}
```

如果需要限制同时支持 IPv4 和 IPv6 寻址的网络访问，可以在 IAM 策略条件中包含 IPv4 和 IPv6 地址或 IP 地址范围。如果用户从指定的 IPv4 或 IPv6 地址范围进行调用，以下基于身份的策略将允许用户将对象放入 `amzn-s3-demo-bucket3` Amazon S3 存储桶。在 IAM 策略中包含 IPv6 地址范围之前，请验证您正在使用的 Amazon Web Services 服务 是否支持 IPv6。有关支持 IPv6 的 Amazon Web Services 服务 列表，请参阅《Amazon VPC 用户指南》中支持 IPv6 的 Amazon Web Services 服务。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "PrincipalPutObjectIfIpAddress",
            "Effect": "Allow",
            "Action": "s3:PutObject",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket3/*",
            "Condition": {
                "IpAddress": {
                    "aws:SourceIp": [
                        "203.0.113.0/24",
                        "2001:DB8:1234:5678::/64"
                    ]
                }
            }
        }
    ]
}
```

如果请求来自使用 Amazon VPC 端点的主机，则 `aws:SourceIp` 密钥不可用。您应改用特定于 VPC 的键，例如 aws:VpcSourceIp。有关使用 VPC 端点的更多信息，请参阅《Amazon PrivateLink 指南》中的 VPC 端点和 VPC 端点服务的身份和访问管理。

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 `Deny` 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

aws:SourceVpc

使用此键检查请求是否经过 VPC 端点附加到的 VPC。在策略中，您可以使用此键以仅允许访问特定的 VPC。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的 限制对特定 VPC 的访问。

- 可用性 – 仅在请求者使用 VPC 端点发出请求时，才将此键包含在请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值

在策略中，可以使用此键来允许或限制对特定 VPC 的访问。

例如，可以将以下基于身份的策略附加到 IAM 角色，以拒绝 `PutObject` 访问 `amzn-s3-demo-bucket3` Amazon S3 存储桶，除非请求是从指定的 VPC ID 发出的，或由使用转发访问会话（FAS）代表角色发出请求的 Amazon Web Services 服务 发出的。与 aws:SourceIp 不同的是，必须使用 aws:ViaAWSService 或 aws:CalledVia 来允许 FAS 请求，因为初始请求的源 VPC 不会被保留。

> **Note**
>
> 该策略不允许进行任何操作。可将此策略与允许特定操作的其他策略结合使用。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "PutObjectIfNotVPCID",
```

```
      "Effect": "Deny",
      "Action": "s3:PutObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket3/*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceVpc": "vpc-1234567890abcdef0"
        },
        "Bool": {
          "aws:ViaAWSService": "false"
        }
      }
    }
  ]
}
```

> **Note**
>
> 如果您的目标服务支持 `aws:SourceVpcArn`，Amazon 建议使用 `aws:SourceVpcArn` 而不是 `aws:SourceVpc`。有关支持的服务列表，请参阅 aws:SourceVpcArn。

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 `Deny` 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

aws:SourceVpcArn

使用此键可验证通过 VPC 端点发出请求的 VPC 的 ARN。此键返回 VPC 端点所连接 VPC 的 ARN。

- 可用性：通过 VPC 端点发出请求时将此键包含在所支持服务的请求上下文中。通过公共服务端点发出的请求不包括此键。以下服务支持此键：
  - Amazon App Runner（前缀：`apprunner`）
  - Amazon Application Discovery Service（前缀：`discovery`）

- Amazon Athena（前缀：athena）
- Amazon Cloud Map（前缀：servicediscovery）
- Amazon CloudWatch Application Insights（前缀：applicationinsights）
- Amazon CloudFormation（前缀：cloudformation）
- Amazon Comprehend Medical（前缀：comprehendmedical）
- Amazon Compute Optimizer（前缀：compute-optimizer）
- Amazon；Elastic Container Registry（前缀：ecr）
- Amazon Elastic Container Service（前缀：ecs）
- Amazon Kinesis Analytics（前缀：kinesisanalytics）
- Amazon Route 53（前缀：route53）
- Amazon DataSync（前缀：datasync）
- Amazon Elastic Block Store（前缀：ebs）
- Amazon EventBridge 调度器（前缀：scheduler）
- Amazon Data Firehose（前缀：firehose）
- Amazon HealthImaging（前缀：medical-imaging）
- Amazon HealthLake（前缀：healthlake）
- Amazon HealthOmics（前缀：omics）
- Amazon Identity and Access Management（`iam:PassRole` 操作除外）（前缀：iam）
- Amazon IoT FleetWise（前缀：iotfleetwise）
- Amazon IoT Wireless（前缀：iotwireless）
- Amazon Key Management Service（前缀：kms）
- Amazon Lambda（前缀：lambda）
- Amazon Payment Cryptography（前缀：payment-cryptography）
- Amazon Polly（前缀：polly）
- Amazon 私有证书颁发机构（前缀：acm-pca）
- Amazon 回收站（前缀：rbin）
- Amazon Rekognition（前缀：rekognition）
- 服务配额（前缀：servicequotas）
- Amazon Simple Storage Service（前缀：s3）
- Amazon Storage Gateway（前缀：storagegateway）

  - Amazon Systems Manager Incident Manager Contacts（前缀：`ssm-contacts`）
  - Amazon Textract（前缀：`textract`）
  - Amazon Transcribe（前缀：`transcribe`）
  - Amazon Transfer Family（前缀：`transfer`）
- 数据类型 – ARN

  Amazon 建议在比较 ARN 时使用 ARN 运算符而不是字符串运算符。
- 值类型 - 单值
- 示例值 – `arn:aws:ec2:us-east-1:123456789012:vpc/vpc-0e9801d129EXAMPLE`

以下是拒绝从 VPC `vpc-1a2b3c4d` 以外的任意位置访问 `amzn-s3-demo-bucket` 及其对象的存储桶策略示例。

```
{
   "Version":"2012-10-17",
   "Statement": [
     {
       "Sid": "Access-to-specific-VPC-only",
       "Principal": "*",
       "Action": "s3:*",
       "Effect": "Deny",
       "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket",
                    "arn:aws:s3:::amzn-s3-demo-bucket/*"],
       "Condition": {
         "ArnNotEquals": {
           "aws:SourceVpcArn": "arn:aws:ec2:us-east-1:*:vpc/vpc-1a2b3c4d"
         }
       }
     }
   ]
}
```

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 Deny 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让

Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

aws:SourceVpce

使用此键可将请求的 VPC 端点标识符与您在策略中指定的端点 ID 进行比较。

- 可用性 – 仅在请求者使用 VPC 端点发出请求时，才将此键包含在请求上下文中。
- 数据类型 – 字符串
- 值类型 - 单值

在策略中，您可以使用此键来限制对特定 VPC 端点的访问。有关更多信息，请参阅《Amazon Simple Storage Service 用户指南》中的限制对特定 VPC 的访问。与使用 aws:SourceVpc 类似，必须使用 aws:ViaAWSService 或 aws:CalledVia 来允许 Amazon Web Services 服务 使用转发访问会话（FAS）发出的请求。这是因为初始请求的源 VPC 端点未被保留。

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 `Deny` 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

aws:VpceAccount

此键用于将发出请求经由的 VPC 端点所属的 Amazon 账户 ID 与您在策略中指定的账户 ID 进行比较。此条件键可确保请求经由特定账户拥有的 VPC 端点，从而帮助建立网络边界控制。

- 可用性 – 通过 VPC 端点发出请求时将此键包含在请求上下文中。通过公共服务端点发出的请求不包括此键。

  以下服务支持此键：

  - Amazon App Runner（前缀：`apprunner`）
  - Amazon Application Discovery Service（前缀：`discovery`）
  - Amazon Athena（前缀：`athena`）

- Amazon Cloud Map（前缀：servicediscovery）
- Amazon CloudWatch Application Insights（前缀：applicationinsights）
- Amazon CloudFormation（前缀：cloudformation）
- Amazon Comprehend Medical（前缀：comprehendmedical）
- Amazon Compute Optimizer（前缀：compute-optimizer）
- Amazon；Elastic Container Registry（前缀：ecr）
- Amazon Elastic Container Service（前缀：ecs）
- Amazon Kinesis Analytics（前缀：kinesisanalytics）
- Amazon Route 53（前缀：route53）
- Amazon DataSync（前缀：datasync）
- Amazon Elastic Block Store（前缀：ebs）
- Amazon EventBridge 调度器（前缀：scheduler）
- Amazon Data Firehose（前缀：firehose）
- Amazon HealthImaging（前缀：medical-imaging）
- Amazon HealthLake（前缀：healthlake）
- Amazon HealthOmics（前缀：omics）
- Amazon Identity and Access Management（iam:PassRole 操作除外）（前缀：iam）
- Amazon IoT FleetWise（前缀：iotfleetwise）
- Amazon IoT Wireless（前缀：iotwireless）
- Amazon Key Management Service（前缀：kms）
- Amazon Lambda（前缀：lambda）
- Amazon Payment Cryptography（前缀：payment-cryptography）
- Amazon Polly（前缀：polly）
- Amazon 私有证书颁发机构（前缀：acm-pca）
- Amazon 回收站（前缀：rbin）
- Amazon Rekognition（前缀：rekognition）
- 服务配额（前缀：servicequotas）
- Amazon Simple Storage Service（前缀：s3）
- Amazon Storage Gateway（前缀：storagegateway）
- Amazon Systems Manager Incident Manager Contacts（前缀：ssm-contacts）

- Amazon Textract（前缀：textract）
- Amazon Transcribe（前缀：transcribe）
- Amazon Transfer Family（前缀：transfer）

- 数据类型 – 字符串
- 值类型 - 单值
- 示例值 – 123456789012

可以使用此条件键来限制对资源的访问，以使请求必须经由您的账户拥有的 VPC 端点。以下 Amazon S3 存储桶策略示例允许在请求经由指定账户拥有的 VPC 端点时进行访问：

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AccessToSpecificVpceAccountOnly",
            "Principal": {
                "AWS": "arn:aws:iam::111122223333:role/RoleName"
            },
            "Action": "s3:GetObject",
            "Effect": "Allow",
            "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
            "Condition": {
                "StringEquals": {
                    "aws:VpceAccount": "111122223333"
                }
            }
        }
    ]
}
```

> **Note**
>
> 此条件键目前适用于一组选定的 Amazon 服务。将条件键用于不支持的服务可能会导致意想不到的授权结果。务必将条件键的范围限定为策略中支持的服务。

某些 Amazon 服务在代表您行事时从其网络访问您的资源。如果使用此等服务，需要对上述策略示例进行编辑，以允许 Amazon 服务从网络外部访问您的资源。有关在基于请求来源强制实施访问控制时需要考虑的访问模式的更多信息，请参阅使用数据边界建立权限防护机制。

aws:VpceOrgID

此键用于将发出请求经由的 VPC 端点所属的 Amazon Organizations 中组织的标识符与您在策略中指定的标识符进行比较。此条件键提供了最具扩展性的网络边界控制方法，自动包括组织内账户拥有的所有 VPC 端点。

- 可用性 - 在通过 VPC 端点发出请求且 VPC 端点的所有者账户是 Amazon 组织成员时，请求上下文中将包含此键。对于通过其他网络路径发出的请求，或者当 VPC 端点的所有者账户不属于某个组织时，则不包括此键。

  以下服务支持此键：

  - Amazon App Runner（前缀：apprunner）
  - Amazon Application Discovery Service（前缀：discovery）
  - Amazon Athena（前缀：athena）
  - Amazon B2B Data Interchange（前缀：b2bi）
  - Amazon Cloud Map（前缀：servicediscovery）
  - Amazon CloudWatch Application Insights（前缀：applicationinsights）
  - Amazon CloudFormation（前缀：cloudformation）
  - Amazon Cognito（前缀：cognito-identity）
  - Amazon Comprehend Medical（前缀：comprehendmedical）
  - Amazon Compute Optimizer（前缀：compute-optimizer）
  - Amazon Database Migration Service（前缀：dms）
  - Amazon 目录服务数据（前缀：ds-data）
  - Amazon；Elastic Container Registry（前缀：ecr）
  - Amazon Elastic Container Service（前缀：ecs）
  - Amazon Kinesis Analytics（前缀：kinesisanalytics）
  - Amazon Route 53（前缀：route53）
  - Amazon DataSync（前缀：datasync）
  - Amazon Elastic Block Store（前缀：ebs）
  - Amazon EventBridge 调度器（前缀：scheduler）

- Amazon Data Firehose（前缀：firehose）
- Amazon HealthImaging（前缀：medical-imaging）
- Amazon HealthLake（前缀：healthlake）
- Amazon HealthOmics（前缀：omics）
- Amazon Identity and Access Management（iam:PassRole 操作除外）（前缀：iam）
- Amazon 身份存储（前缀：identitystore）
- Amazon IoT FleetWise（前缀：iotfleetwise）
- Amazon IoT TwinMaker（前缀：iottwinmaker）
- Amazon IoT Wireless（前缀：iotwireless）
- Amazon Keyspaces（Apache Cassandra 兼容）（前缀：cassandra）
- Amazon Key Management Service（前缀：kms）
- Amazon Lambda（前缀：lambda）
- Amazon Network Firewall（前缀：network-firewall）
- Amazon Payment Cryptography（前缀：payment-cryptography）
- Amazon Pinpoint SMS and Voice Service（前缀：sms-voice）
- Amazon Polly（前缀：polly）
- Amazon 价目表（前缀：pricing）
- Amazon 私有证书颁发机构（前缀：acm-pca）
- Amazon 回收站（前缀：rbin）
- Amazon Rekognition（前缀：rekognition）
- 服务配额（前缀：servicequotas）
- Amazon Simple Email Service（前缀：ses）
- Amazon Simple Storage Service（前缀：s3）
- Amazon Simple Queue Service（前缀：sqs）
- Amazon Storage Gateway（前缀：storagegateway）
- Amazon Systems Manager Incident Manager Contacts（前缀：ssm-contacts）
- Amazon Textract（前缀：textract）
- Amazon Transcribe（前缀：transcribe）
- Amazon Transfer Family（前缀：transfer）
- Amazon WorkMail（前缀：workmail）

- 数据类型 – 字符串
- 值类型 - 单值
- 示例值 – o-a1b2c3d4e5

以下资源控制策略示例拒绝访问 Amazon S3 和 Amazon Key Management Service 资源，除非请求经由指定组织拥有的 VPC 端点或代表您行事的 Amazon 服务网络。某些组织可能需要进一步编辑此政策以满足其组织的需求，例如，允许第三方合作伙伴进行访问。有关在基于请求来源强制实施访问控制时需要考虑的访问模式的更多信息，请参阅使用数据边界建立权限防护机制。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "EnforceNetworkPerimeterVpceOrgID",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:*",
        "kms:*"
      ],
      "Resource": "*",
      "Condition": {
        "BoolIfExists": {
          "aws:PrincipalIsAWSService": "false",
          "aws:ViaAWSService": "false"
        },
        "StringNotEqualsIfExists": {
            "aws:VpceOrgID": "o-abcdef0123",
            "aws:PrincipalTag/network-perimeter-exception": "true"
        }
      }
    }
  ]
}
```

> **Note**
>
> 此条件键目前适用于一组选定的 Amazon 服务。将条件键用于不支持的服务可能会导致意想不到的授权结果。务必将条件键的范围限定为策略中支持的服务。

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 `Deny` 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

aws:VpceOrgPaths

此键用于将发出请求经由的 VPC 端点的 Amazon Organizations 路径与您在策略中指定的路径进行比较。此条件键支持在组织单元 (OU) 级别实施网络边界控制，当您在指定的 OU 中添加新的端点时，可根据 VPC 端点的使用情况自动扩展。

- 可用性 - 在通过 VPC 端点发出请求且 VPC 端点的所有者账户是组织成员时，请求上下文中将包含此键。对于通过其他网络路径发出的请求，或者当 VPC 端点的所有者账户不属于某个组织时，则不包括此键。

  以下服务支持此键：

  - Amazon App Runner（前缀：`apprunner`）
  - Amazon Application Discovery Service（前缀：`discovery`）
  - Amazon Athena（前缀：`athena`）
  - Amazon Cloud Map（前缀：`servicediscovery`）
  - Amazon CloudWatch Application Insights（前缀：`applicationinsights`）
  - Amazon CloudFormation（前缀：`cloudformation`）
  - Amazon Comprehend Medical（前缀：`comprehendmedical`）
  - Amazon Compute Optimizer（前缀：`compute-optimizer`）
  - Amazon；Elastic Container Registry（前缀：`ecr`）
  - Amazon Elastic Container Service（前缀：`ecs`）

- Amazon Kinesis Analytics（前缀：kinesisanalytics）
- Amazon Route 53（前缀：route53）
- Amazon DataSync（前缀：datasync）
- Amazon Elastic Block Store（前缀：ebs）
- Amazon EventBridge 调度器（前缀：scheduler）
- Amazon Data Firehose（前缀：firehose）
- Amazon HealthImaging（前缀：medical-imaging）
- Amazon HealthLake（前缀：healthlake）
- Amazon HealthOmics（前缀：omics）
- Amazon Identity and Access Management（iam:PassRole 操作除外）（前缀：iam）
- Amazon IoT FleetWise（前缀：iotfleetwise）
- Amazon IoT Wireless（前缀：iotwireless）
- Amazon Key Management Service（前缀：kms）
- Amazon Lambda（前缀：lambda）
- Amazon Payment Cryptography（前缀：payment-cryptography）
- Amazon Polly（前缀：polly）
- Amazon 私有证书颁发机构（前缀：acm-pca）
- Amazon 回收站（前缀：rbin）
- Amazon Rekognition（前缀：rekognition）
- 服务配额（前缀：servicequotas）
- Amazon Simple Storage Service（前缀：s3）
- Amazon Storage Gateway（前缀：storagegateway）
- Amazon Systems Manager Incident Manager Contacts（前缀：ssm-contacts）
- Amazon Textract（前缀：textract）
- Amazon Transcribe（前缀：transcribe）
- Amazon Transfer Family（前缀：transfer）

- 数据类型 – 字符串（列表）
- 值类型— 多值
- 示例值 – o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/

由于 aws:VpceOrgPaths 是一个多值条件键，您必须将 ForAnyValue 或者 ForAllValues 集合运算符与此键的字符串条件运算符搭配使用。以下 Amazon S3 存储桶策略示例仅允许在请求经由特定组织单元中的账户拥有的 VPC 端点时进行访问：

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "AllowAccessFromSpecificOrgPaths",
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111122223333:role/RoleName"
      },
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::amzn-s3-demo-bucket1/*",
      "Condition": {
        "ForAnyValue:StringLike": {
          "aws:VpceOrgPaths": [
            "o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/*",
            "o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-33333333/*"
          ]
        }
      }
    }
  ]
}
```

> **Note**
>
> 此条件键目前适用于一组选定的 Amazon 服务。将条件键用于不支持的服务可能会导致意想不到的授权结果。务必将条件键的范围限定为策略中支持的服务。

某些 Amazon 服务在代表您行事时从其网络访问您的资源。如果使用此等服务，需要对上述策略示例进行编辑，以允许 Amazon 服务从网络外部访问您的资源。有关在基于请求来源强制实施访问控制时需要考虑的访问模式的更多信息，请参阅使用数据边界建立权限防护机制。

aws:VpcSourceIp

使用此键可将从中发出请求的 IP 地址与您在策略中指定的 IP 地址进行比较。在策略中，键仅在请求来自指定的 IP 地址并经过 VPC 端点时进行匹配。

- 可用性 – 仅在使用 VPC 端点发出请求时，才将此键包含在请求上下文中。
- 数据类型 – IP 地址
- 值类型 - 单值

有关更多信息，请参阅《Amazon VPC 用户指南》中的使用端点策略控制对 VPC 端点的访问。与使用 aws:SourceVpc 类似，必须使用 aws:ViaAWSService 或 aws:CalledVia 来允许 Amazon Web Services 服务 使用转发访问会话（FAS）发出的请求。这是因为使用 VPC 端点发出的初始请求的源 IP 不会保留在 FAS 请求中。

> **Note**
>
> `aws:VpcSourceIp` 同时支持 IPv4 和 IPv6 地址或 IP 地址范围。有关支持 IPv6 的 Amazon Web Services 服务 列表，请参阅《Amazon VPC 用户指南》中支持 IPv6 的 Amazon Web Services 服务。
> 始终将 `aws:VpcSourceIp` 条件键与 `aws:SourceVpc`、`aws:SourceVpce` 或 `aws:SourceVpcArn` 条件键一起使用。否则，某项策略可能会允许来自意外 VPC 的 API 调用，而该 VPC 使用的是相同或重叠的 IP CIDR。出现这种情况是因为两个互不相关的 VPC 的 IP CIDR 可能相同或存在重叠。

> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 `Deny` 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 `false` 的 `aws:PrincipalIsAWSService` 条件键将服务主体排除在 `Deny` 语句之外。

资源的属性

使用以下条件键可将有关请求目标资源的详细信息与您在策略中指定的资源属性进行比较。

aws:ResourceAccount

使用此键，将策略中所请求的资源所有者的 Amazon Web Services 账户 ID 与资源账户进行比较。然后，您可以根据拥有该资源的账户来允许或拒绝对该资源的访问。

- Availability（可用性）– 对于大多数服务操作，此键始终包含在请求上下文中。以下操作不支持此键：
  - Amazon Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Directory Service
    - `ds:AcceptSharedDirectory`
  - Amazon Elastic Block Store – 所有操作
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents` – EventBridge `PutEvents` 将调用其他账户中的事件总线，前提是该事件总线已在 2023 年 3 月 2 日之前被配置为跨账户 EventBridge 目标。有关更多信息，请参阅《Amazon EventBridge 用户指南》中的 授予权限以允许来自其他 Amazon 账户的事件。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`
  - Amazon Macie
    - `macie2:AcceptInvitation`
  - Amazon OpenSearch Service
    - `es:AcceptInboundConnection`
  - Amazon Route 53

    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
  - Amazon Security Hub CSPM
    - `securityhub:AcceptAdministratorInvitation`
- 数据类型 – 字符串
- 值类型 - 单值

> **Note**
>
> 有关上述不支持的操作的其他注意事项，请参阅 Data Perimeter Policy Examples 存储库。

此键等于包含请求中受评估资源的账户的 Amazon Web Services 账户 ID。

对于您的账户中的大多数资源，ARN 包含该资源的所有者账户 ID。对于某些资源（如 Amazon S3 存储桶），资源 ARN 不包含该账户 ID。以下两个示例显示了在 ARN 中具有账户 ID 的资源与没有账户 ID 的 Amazon S3 ARN 之间的区别：

- `arn:aws:iam::123456789012:role/AWSExampleRole` – 在账户 123456789012 中创建和拥有的 IAM 角色。
- `arn:aws:s3:::amzn-s3-demo-bucket2` – 在账户 111122223333 中创建且拥有的 Amazon S3 存储桶，在 ARN 中不显示。

使用 Amazon 控制台、API 或 CLI 查找您的所有资源和相应的 ARN。

您编写的策略根据资源所有者的账户 ID 拒绝对资源的权限。例如，如果指定资源不属于指定账户，以下基于身份的策略会拒绝访问该资源。

要使用此策略，请将示例策略中的斜体占位符文本替换为您的账户信息。

> **Important**
>
> 该策略不允许进行任何操作。相反，该策略会使用 Deny 效果，这会显式拒绝访问语句中列出而不属于已列出账户的所有资源。将此策略与允许访问特定资源的其他策略结合使用。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "DenyInteractionWithResourcesNotInSpecificAccount",
      "Action": "service:*",
      "Effect": "Deny",
      "Resource": [
        "arn:aws:service:us-east-1:111122223333:*"
      ],
      "Condition": {
        "StringNotEquals": {
          "aws:ResourceAccount": [
            "111122223333"
          ]
        }
      }
    }
  ]
}
```

此策略会拒绝对特定 Amazon 服务所有资源的访问，除非指定 Amazon Web Services 账户 拥有该资源。

> **Note**
>
> 某些 Amazon Web Services 服务 需要访问托管在其他 Amazon 中的 Amazon Web Services 账户 所拥有的资源。在基于身份的策略中使用 `aws:ResourceAccount` 可能会影响您的身份访问这些资源的能力。

某些 Amazon 服务，例如 Amazon Web Services Data Exchange，依赖于对您的 Amazon Web Services 账户 以外的资源的访问来进行正常操作。如果您在策略中使用元素 `aws:ResourceAccount`，请在策略中包含额外声明来为这些服务创建豁免。示例策略 Amazon：拒绝访问您账户之外的 Amazon S3 资源，Amazon Web Services Data Exchange 除外。 演示了如何根据资源账户拒绝访问，同时定义对于服务拥有的资源的例外情况。

将此策略示例用作创建自己的自定义策略的模板。参阅服务文档以了解更多信息。

aws:ResourceOrgPaths

使用此键来比较所访问资源的 Amazon Organizations 路径与策略中的路径。在策略中，此条件键可确保该资源属于 Amazon Organizations 中指定组织根或组织单元 (OU) 的账户成员。Amazon Organizations 路径是 Organizations 实体结构的文本表示形式。有关使用和了解路径的更多信息，请参阅 了解 Amazon Organizations 实体路径

- Availability（可用性）- 仅在拥有资源的账户是企业成员时，才将此键包含在请求上下文中。此全局条件键不支持以下操作：
  - Amazon Audit Manager
    - `auditmanager:UpdateAssessmentFrameworkShare`
  - Amazon Detective
    - `detective:AcceptInvitation`
  - Amazon Directory Service
    - `ds:AcceptSharedDirectory`
  - Amazon Elastic Block Store – 所有操作
  - Amazon EC2
    - `ec2:AcceptTransitGatewayPeeringAttachment`
    - `ec2:AcceptVpcEndpointConnections`
    - `ec2:AcceptVpcPeeringConnection`
    - `ec2:CreateTransitGatewayPeeringAttachment`
    - `ec2:CreateVpcEndpoint`
    - `ec2:CreateVpcPeeringConnection`
  - Amazon EventBridge
    - `events:PutEvents` – EventBridge `PutEvents` 将调用其他账户中的事件总线，前提是该事件总线已在 2023 年 3 月 2 日之前被配置为跨账户 EventBridge 目标。有关更多信息，请参阅《Amazon EventBridge 用户指南》中的 授予权限以允许来自其他 Amazon 账户的事件。
  - Amazon GuardDuty
    - `guardduty:AcceptAdministratorInvitation`
  - Amazon Macie
    - `macie2:AcceptInvitation`

  - Amazon OpenSearch Service
    - `es:AcceptInboundConnection`
  - Amazon Route 53
    - `route53:AssociateVpcWithHostedZone`
    - `route53:CreateVPCAssociationAuthorization`
  - Amazon Security Hub CSPM
    - `securityhub:AcceptAdministratorInvitation`
- 数据类型 – 字符串（列表）
- 值类型— 多值

> **Note**
>
> 有关上述不支持的操作的其他注意事项，请参阅 Data Perimeter Policy Examples 存储库。

`aws:ResourceOrgPaths` 是一个多值条件键。多值键在请求上下文中可以有多个值。对于此键，您必须将 `ForAnyValue` 或者 `ForAllValues` 集合运算符与字符串条件运算符搭配使用。有关多值条件键的更多信息，请参阅多值上下文键的集合运算符。

例如，对于属于组织 `o-a1b2c3d4e5` 的资源，以下条件会返回为 `True`。包含通配符时，您必须使用 StringLike 条件运算符。

```
"Condition": {
      "ForAnyValue:StringLike": {
            "aws:ResourceOrgPaths":["o-a1b2c3d4e5/*"]
   }
}
```

对于具有 OU ID `ou-ab12-11111111` 的资源，以下条件返回 `True`。它将匹配附加到 OU ou-ab12-11111111 或任何子 OU 的账户所拥有的资源。

```
"Condition": { "ForAnyValue:StringLike" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/*"]
}}
```

对于直接附加到 OU ID ou-ab12-22222222（而非子 OU）的账户所拥有的资源，以下条件返回 True。下列示例使用 StringEquals 条件运算符来指定 OU ID 的完全匹配（而非通配符匹配）要求。

```
"Condition": { "ForAnyValue:StringEquals" : {
     "aws:ResourceOrgPaths":["o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-ab12-22222222/"]
}}
```

> **Note**
>
> 某些 Amazon Web Services 服务 需要访问托管在其他 Amazon 中的 Amazon Web Services 账户 所拥有的资源。在基于身份的策略中使用 aws:ResourceOrgPaths 可能会影响您的身份访问这些资源的能力。

某些 Amazon 服务，例如 Amazon Web Services Data Exchange，依赖于对您的 Amazon Web Services 账户 以外的资源的访问来进行正常操作。如果您在策略中使用 aws:ResourceOrgPaths 键，请在策略中包含额外声明来为这些服务创建豁免。示例策略 Amazon：拒绝访问您账户之外的 Amazon S3 资源，Amazon Web Services Data Exchange 除外。 演示了如何根据资源账户拒绝访问，同时定义对于服务拥有的资源的例外情况。您可以使用 aws:ResourceOrgPaths 键来创建类似的策略，限制对组织单元（OU）内部资源的访问，同时包括服务拥有的资源。

将此策略示例用作创建自己的自定义策略的模板。参阅服务文档以了解更多信息。

aws:ResourceOrgID

使用此键可将被请求资源所属的 Amazon Organizations 中组织的标识符与策略中指定的标识符进行比较。

- Availability（可用性）- 仅在拥有资源的账户是企业成员时，才将此键包含在请求上下文中。此全局条件键不支持以下操作：
  - Amazon Audit Manager
    - auditmanager:UpdateAssessmentFrameworkShare
  - Amazon Detective
    - detective:AcceptInvitation
  - Amazon Directory Service
    - ds:AcceptSharedDirectory
  - Amazon Elastic Block Store – 所有操作

- Amazon EC2
  - `ec2:AcceptTransitGatewayPeeringAttachment`
  - `ec2:AcceptVpcEndpointConnections`
  - `ec2:AcceptVpcPeeringConnection`
  - `ec2:CreateTransitGatewayPeeringAttachment`
  - `ec2:CreateVpcEndpoint`
  - `ec2:CreateVpcPeeringConnection`
- Amazon EventBridge
  - `events:PutEvents` – EventBridge `PutEvents` 将调用其他账户中的事件总线，前提是该事件总线已在 2023 年 3 月 2 日之前被配置为跨账户 EventBridge 目标。有关更多信息，请参阅《Amazon EventBridge 用户指南》中的 授予权限以允许来自其他 Amazon 账户的事件。
- Amazon GuardDuty
  - `guardduty:AcceptAdministratorInvitation`
- Amazon Macie
  - `macie2:AcceptInvitation`
- Amazon OpenSearch Service
  - `es:AcceptInboundConnection`
- Amazon Route 53
  - `route53:AssociateVpcWithHostedZone`
  - `route53:CreateVPCAssociationAuthorization`
- Amazon Security Hub CSPM
  - `securityhub:AcceptAdministratorInvitation`
- 数据类型 – 字符串
- 值类型 - 单值

> **Note**
>
> 有关上述不支持的操作的其他注意事项，请参阅 Data Perimeter Policy Examples 存储库。

此全局键会返回给定请求的资源组织 ID。它允许您创建适用于组织中所有资源的规则，这些资源在基于身份的策略的 Resource 元素中指定。您可以在条件元素中指定组织 ID。当您添加和删除账户时，包含 aws:ResourceOrgID 键的策略将自动包括正确的账户，并且不必手动更新。

例如，以下策略会阻止主体将对象添加到 policy-genius-dev 资源，除非 Amazon S3 资源与发出请求的主体属于同一组织。

> **Important**
>
> 该策略不允许进行任何操作。相反，该策略会使用 Deny 效果，这会显式拒绝访问语句中列出而不属于已列出账户的所有资源。将此策略与允许访问特定资源的其他策略结合使用。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": {
        "Sid": "DenyPutObjectToS3ResourcesOutsideMyOrganization",
        "Effect": "Deny",
        "Action": "s3:PutObject",
        "Resource": "arn:aws:s3:::policy-genius-dev/*",
        "Condition": {
            "StringNotEquals": {
                "aws:ResourceOrgID": "${aws:PrincipalOrgID}"
            }
        }
    }
}
```

> **Note**
>
> 某些 Amazon Web Services 服务 需要访问托管在其他 Amazon 中的 Amazon Web Services 账户 所拥有的资源。在基于身份的策略中使用 aws:ResourceOrgID 可能会影响您的身份访问这些资源的能力。

某些 Amazon 服务，例如 Amazon Web Services Data Exchange，依赖于对您的 Amazon Web Services 账户 以外的资源的访问来进行正常操作。如果您在策略中使用 aws:ResourceOrgID 键，

请在策略中包含额外声明来为这些服务创建豁免。示例策略 Amazon：拒绝访问您账户之外的 Amazon S3 资源，Amazon Web Services Data Exchange 除外。 演示了如何根据资源账户拒绝访问，同时定义对于服务拥有的资源的例外情况。您可以使用 aws:ResourceOrgID 键来创建类似的策略，限制对企业内部资源的访问，同时包括服务拥有的资源。

将此策略示例用作创建自己的自定义策略的模板。参阅服务文档以了解更多信息。

在以下视频中，了解有关如何在策略中使用 aws:ResourceOrgID 条件键的更多信息。

确保身份和网络只能用于访问受信任的资源。

aws:ResourceTag/tag-key

使用此键可将您在策略中指定的标签键/值对与附加到资源的键/值对进行比较。例如，您可能会要求只有在资源具有附加的标签键 "Dept" 和值 "Marketing" 时才允许访问该资源。有关更多信息，请参阅 控制对 Amazon 资源的访问。

- Availability（可用性）– 当请求的资源已附加标签，或者是在利用附加包创建资源的请求中，键会包含在请求上下文中。只有支持基于标签的授权的资源返回此键。每个标签键/值对均有一个上下文密钥。
- 数据类型 – 字符串
- 值类型— 单值

该上下文密钥的格式为 "aws:ResourceTag/*tag-key*":"*tag-value*"，其中 *tag-key* 和 *tag-value* 是标签键值对。标签键不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的资源标签键，但不会同时匹配两者。这些标签键/值对中的值应区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:ResourceTag/TagKey1": "Production"，则条件将匹配名为 Production 的资源标签值，但不会同时匹配 production 或 PRODUCTION。

有关使用 aws:ResourceTag 键控制对 IAM 资源的访问的示例，请参阅 控制对 Amazon 资源的访问。

有关使用 aws:ResourceTag 键控制对其他 Amazon 资源的访问的示例，请参阅 使用标签控制对 Amazon 资源的访问。

有关使用 aws:ResourceTag 条件键进行基于属性的访问权限控制 (ABAC) 的教程，请参阅 IAM 教程：根据标签定义访问 Amazon 资源的权限。

请求的属性

使用以下条件键可将有关请求本身以及请求内容的详细信息与您在策略中指定的请求属性进行比较。

aws:CalledVia

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的服务进行比较。当主体向 Amazon 服务发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。当使用前向访问会话 (FAS) 发出请求时，此键将设置为服务主体的值。`aws:CalledVia` 键包含链中代表主体发出请求的每个服务的有序列表。

有关更多信息，请参阅 转发访问会话。

- 可用性 - 当支持 `aws:CalledVia` 的服务使用 IAM 主体的凭证向其他服务发出请求时，此键会出现在请求中。如果服务使用服务角色或服务相关角色代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 数据类型 – 字符串（列表）
- 值类型— 多值

要在策略中使用 `aws:CalledVia` 条件键，您必须提供服务主体以允许或拒绝 Amazon 服务请求。例如，您可以使用 Amazon CloudFormation 从 Amazon DynamoDB 表中读取和写入。然后，DynamoDB 使用 Amazon Key Management Service (Amazon KMS) 提供的加密操作。

要在任何 服务使用主体的凭证发出请求时允许或拒绝访问，请使用 `aws:ViaAWSService` 条件键。该条件键支持 Amazon 服务。

`aws:CalledVia` 键是多值键。但是，您不能在条件下强制使用此键进行排序。使用上面的示例，User 1（用户 1）向 Amazon CloudFormation 发出请求，然后依次调用 DynamoDB 和调用 Amazon KMS。这是三个单独的请求。对 Amazon KMS 的最终调用是由用户 1 通过 Amazon CloudFormation，然后通过 DynamoDB 进行的。
使用 aws:CalledVia 的示例。

在这种情况下，请求上下文中的 `aws:CalledVia` 键包括 `cloudformation.amazonaws.com` 和 `dynamodb.amazonaws.com`（按照该顺序）。如果您只关心调用是在请求链中的某个地方通过 DynamoDB 进行的，则可以在策略中使用此条件键。

例如，以下策略允许管理名为 `my-example-key` 的 Amazon KMS 键，但前提为 DynamoDB 是发出请求的服务之一。`ForAnyValue:StringEquals` 条件运算符确保 DynamoDB 是发出调用的服务之一。如果主体直接调用 Amazon KMS，则条件将返回 `false`，且此策略不允许请求。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaDynamodb",
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/my-example-key",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "aws:CalledVia": [
                        "dynamodb.amazonaws.com"
                    ]
                }
            }
        }
    ]
}
```

如果要强制由哪个服务在链中进行第一个或最后一个调用，可以使用 aws:CalledViaFirst 和 aws:CalledViaLast 键。例如，以下策略允许管理 Amazon KMS 中名为 my-example-key 的键。仅当链中包含多个请求时，才允许执行这些 Amazon KMS 操作。第一个请求必须通过 Amazon CloudFormation 发出，而最后一个请求必须通过 DynamoDB 发出。如果其他服务在链的中间发出请求，则仍然允许该操作。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "KmsActionsIfCalledViaChain",
```

```
            "Effect": "Allow",
            "Action": [
                "kms:Encrypt",
                "kms:Decrypt",
                "kms:ReEncrypt*",
                "kms:GenerateDataKey",
                "kms:DescribeKey"
            ],
            "Resource": "arn:aws:kms:us-east-1:111122223333:key/my-example-key",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaFirst": "cloudformation.amazonaws.com",
                    "aws:CalledViaLast": "dynamodb.amazonaws.com"
                }
            }
        }
    ]
}
```

当服务使用 IAM 主体的凭证调用另一个服务时，请求中会存在 aws:CalledViaFirst 和 aws:CalledViaLast 键。它们表示在请求链中进行调用的第一个和最后一个服务。例如，假定 Amazon CloudFormation 调用另一个名为 X Service 的服务，然后依次调用 DynamoDB，然后调用 Amazon KMS。对 Amazon KMS 的最终调用是由 User 1 依次通过 Amazon CloudFormation、X Service 和 DynamoDB 进行的。首先通过 DynamoDB 进行调用，最后通过 Amazon CloudFormation 进行调用。
使用 aws:CalledViaFirst 和 aws:CalledViaLast 的示例。

aws:CalledViaFirst

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的第一个服务 进行比较。有关更多信息，请参阅 aws:CalledVia。

- Availability（可用性）- 当服务使用 IAM 主体的凭证向其他服务发出至少一个其他请求时，此键会出现在请求中。如果服务使用服务角色或服务相关角色代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 数据类型 – 字符串
- 值类型 - 单值

aws:CalledViaLast

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的最后一个服务进行比较。有关更多信息，请参阅 aws:CalledVia。

- Availability（可用性）- 当服务使用 IAM 主体的凭证向其他服务发出至少一个其他请求时，此键会出现在请求中。如果服务使用服务角色或服务相关角色代表主体进行调用，则此键不存在。当主体直接进行调用时，此键也不存在。
- 数据类型 – 字符串
- 值类型 - 单值

aws:ViaAWSService

使用此键检查 Amazon Web Services 服务 是否使用转发访问会话（FAS）代表你向另一项服务发出请求。

当服务使用转发访问会话代表原始 IAM 主体发出请求时，请求上下文键会返回 `true`。当主体直接进行调用时，请求上下文密钥也会返回 `false`。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 布尔值
- 值类型 - 单值

aws:CalledViaAWSMCP

使用此键可以将策略中的服务与代表 IAM 主体（用户或角色）发出请求的 Amazon MCP 服务进行比较。当主体向 Amazon MCP 服务发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。当使用 Amazon MCP 服务发出请求时，此键将设置为服务主体的值。`aws:CalledViaAWSMCP` 键包含代表主体发出请求的 MCP 服务的服务主体名称。

- 可用性 - 当 Amazon MCP 服务使用 IAM 主体的凭证向 Amazon 服务发出请求时，此键会出现在请求中。当主体直接进行调用时，此键也不存在。
- 数据类型 – 字符串
- 值类型 - 单值

您可以根据由哪个特定 MCP 服务器发出请求，使用此条件键允许或拒绝访问。例如，当敏感的删除操作通过特定 MCP 服务器发出时，以下策略会拒绝这些操作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenySensitiveActionsViaSpecificMCP",
            "Effect": "Deny",
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteObject",
                "dynamodb:DeleteTable"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:CalledViaAWSMCP": "aws-mcp.amazonaws.com"
                }
            }
        }
    ]
}
```

aws:ViaAWSMCPService

使用此键检查 Amazon MCP 服务是否使用转发访问会话（FAS）代表你向另一项 Amazon 服务发出请求。当 Amazon MCP 服务代表原始 IAM 主体将请求转发到 Amazon 请求时，请求上下文键会返回 `true`。当主体直接进行调用时，请求上下文密钥也会返回 `false`。

- 可用性 – 当 Amazon MCP 服务器代表 IAM 主体向下游 Amazon 服务发出请求时，此键包含在请求上下文中。
- 数据类型 – 布尔值
- 值类型 - 单值

当特定操作通过 MCP 服务器时，您可以使用此键来限制这些操作。例如，以下策略拒绝通过任何 Amazon MCP 服务器发出的敏感删除操作：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "DenySensitiveActionsViaMCP",
```

```
            "Effect": "Deny",
            "Action": [
                "s3:DeleteBucket",
                "s3:DeleteObject",
                "dynamodb:DeleteTable"
            ],
            "Resource": "*",
            "Condition": {
                "Bool": {
                    "aws:ViaAWSMCPService": "true"
                }
            }
        }
    ]
}
```

aws:CurrentTime

使用此键可将请求的日期和时间与您在策略中指定的日期和时间进行比较。要查看使用此条件键的示例策略，请参阅Amazon：允许基于日期和时间进行访问。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 日期
- 值类型 - 单值

aws:EpochTime

使用此键可将请求的日期和时间（以 epoch 或 Unix 时间表示）与您在策略中指定的值进行比较。此键还接受自 1970 年 1 月 1 日以来的秒数。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 日期、数字
- 值类型 - 单值

aws:referer

使用此键可将在客户端浏览器中引用请求的站点与您在策略中指定的引用站点进行比较。aws:referer 请求上下文值是由调用方在 HTTP 标头中提供的。当您在网页上选择链接时，Referer 标题将包含在 Web 浏览器请求中。Referer 标题包含选定链接的网页的 URL。

- Availability（可用性）- 仅当通过从浏览器中的网页 URL 链接来调用对 Amazon 资源的请求时，才将此键包含在请求上下文中。编程请求不包括此键，因为它不使用浏览器链接即可访问 Amazon 资源。
- 数据类型 – 字符串
- 值类型— 单值

例如，您可以直接使用 URL 或使用直接 API 调用访问 Amazon S3 对象。有关更多信息，请参阅使用 Web 浏览器直接调用 Amazon S3 API 操作。当您从网页中存在的 URL 访问 Amazon S3 对象时，源网页的 URL 将在 `aws:referer` 中使用。当您通过在浏览器中键入 URL 来访问 Amazon S3 对象时，`aws:referer` 不存在。直接调用 API 时，`aws:referer` 也不存在。您可以使用策略中的 `aws:referer` 条件键以允许来自特定引用站点发出的请求，例如公司域中网页上的链接。

> **Warning**
>
> 应谨慎使用此键。包含公共已知的引用站点标头值是非常危险的。未经授权方可能会使用修改的浏览器或自定义浏览器提供他们选择的任何 `aws:referer` 值。因此，`aws:referer` 不应用于阻止未经授权方直接发出 Amazon 请求。提供它只是为了允许客户保护其数字内容（如存储在 Amazon S3 中的内容），以免在未经授权的第三方站点上引用。

aws:RequestedRegion

使用此键可将已在请求中调用的 Amazon 区域与您在策略中指定的区域进行比较。您可以使用此全局条件键来控制可请求的区域。要查看每个服务的 Amazon 区域，请参阅《Amazon Web Services 一般参考》中的 服务端点和限额。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 字符串
- 值类型— 单值

一些全球服务（如 IAM）具有单个端点。因为此端点的物理位置美国东部（弗吉尼亚州北部）区域，所以 IAM 调用始终会向 us-east-1 区域发出。例如，如果您创建一个策略以在请求的区域不是 us-west-2 时拒绝访问所有服务，则 IAM 调用将始终失败。要查看如何解决此问题的示例，请参阅 NotAction 与 Deny。

> **Note**
>
> 使用 aws:RequestedRegion 条件键可以控制服务的哪个端点被调用，但不控制该操作产生的影响。一些服务具有跨区域影响。
> 例如，Amazon S3 具有跨区域扩展的 API 操作。
>
> - 您可以在一个区域中调用 s3:PutBucketReplication（受 aws:RequestedRegion 条件键的影响），但其他区域将会受到影响，具体取决于复制配置设置。
> - 您可以调用 s3:CreateBucket 在另一个区域创建存储桶，然后使用 s3:LocationConstraint 条件键来控制适用区域。

您可以使用该上下文密钥将访问限制为一组给定区域中的 Amazon 服务。例如，以下策略允许用户查看 Amazon Web Services 管理控制台 中的所有 Amazon EC2 实例。但是，它仅允许用户更改爱尔兰 (eu-west-1)、伦敦 (eu-west-2) 或巴黎 (eu-west-3) 中的实例。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "InstanceConsoleReadOnly",
            "Effect": "Allow",
            "Action": [
                "ec2:Describe*",
                "ec2:Export*",
                "ec2:Get*",
                "ec2:Search*"
            ],
            "Resource": "*"
        },
        {
            "Sid": "InstanceWriteRegionRestricted",
            "Effect": "Allow",
            "Action": [
                "ec2:Associate*",
                "ec2:Import*",
                "ec2:Modify*",
                "ec2:Monitor*",
```

```
                "ec2:Reset*",
                "ec2:Run*",
                "ec2:Start*",
                "ec2:Stop*",
                "ec2:Terminate*"
            ],
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "aws:RequestedRegion": [
                        "eu-west-1",
                        "eu-west-2",
                        "eu-west-3"
                    ]
                }
            }
        }
    ]
}
```

aws:RequestTag/tag-key

使用此键可将请求中传递的标签键/值对与您在策略中指定的标签对进行比较。例如，您可以检查请求是否包含标签键 "Dept" 并具有 "Accounting" 值。有关更多信息，请参阅 在 Amazon 请求期间控制访问。

- Availability（可用性）– 当在请求中传递标签键值对时，将此键包含在请求上下文中。在请求中传递多个标签时，每个标签键/值对均有一个上下文密钥。
- 数据类型 – 字符串
- 值类型— 单值

该上下文密钥的格式为 "aws:RequestTag/*tag-key*":"*tag-value*"，其中 *tag-key* 和 *tag-value* 是标签键值对。标签键不区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:RequestTag/TagKey1": "Value1"，则条件将匹配名为 TagKey1 或 tagkey1 的请求标签键，但不会同时匹配两者。这些标签键/值对中的值应区分大小写。这意味着，如果您在策略的条件元素中指定 "aws:RequestTag/TagKey1": "Production"，则条件将匹配名为 Production 的请求标签值，但不会同时匹配 production 或 PRODUCTION。

这个例子表明，虽然该键是单值的，但如果键不同，您仍然可以在请求中使用多个键值对。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Effect": "Allow",
    "Action": "ec2:CreateTags",
    "Resource": "arn:aws:ec2::111122223333:instance/*",
    "Condition": {
      "StringEquals": {
        "aws:RequestTag/environment": [
          "preprod",
          "production"
        ],
        "aws:RequestTag/team": [
          "engineering"
        ]
      }
    }
  }
}
```

aws:TagKeys

使用此键可将请求中的标签键与您在策略中指定的键进行比较。我们建议当使用策略来通过标签控制访问时，请使用 aws:TagKeys 条件键来定义允许的标签键。有关示例策略和更多信息，请参阅the section called “根据标签键控制访问”。

- Availability（可用性）– 仅在操作支持将标签传递到请求中时，才将此键包含在请求上下文中。
- 数据类型 – 字符串（列表）
- 值类型— 多值

该上下文密钥的格式为 "aws:TagKeys":"*tag-key*"，其中 *tag-key* 是没有值的标签键列表 (例如，["Dept","Cost-Center"])。

由于可在一个请求中包含多个标签键/值对，因此，请求内容可以是多值请求。在此情况下，您必须使用 ForAllValues 或 ForAnyValue 集合运算符。有关更多信息，请参阅 多值上下文键的集合运算符。

某些服务支持同时使用标记和资源操作，例如创建、修改或删除资源。要允许在单次调用中同时使用标记和操作，您必须创建同时包含标记操作和资源修改操作的策略。然后，您可以使用 aws:TagKeys 条件键强制在请求中使用特定标签键。例如，要在某人创建 Amazon EC2 快照时限制标签，您必须在策略中包含 ec2:CreateSnapshot创建操作和 ec2:CreateTags 标记操作。要查看此使用 aws:TagKeys 场景的策略，请参阅《Amazon EC2 用户指南》中的创建具有标签的快照。

aws:SecureTransport

使用此键可检查是否已使用 TLS 发送请求。请求上下文将返回 true 或 false。在策略中，只能在使用 TLS 发送请求时允许特定操作。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 布尔值
- 值类型 - 单值

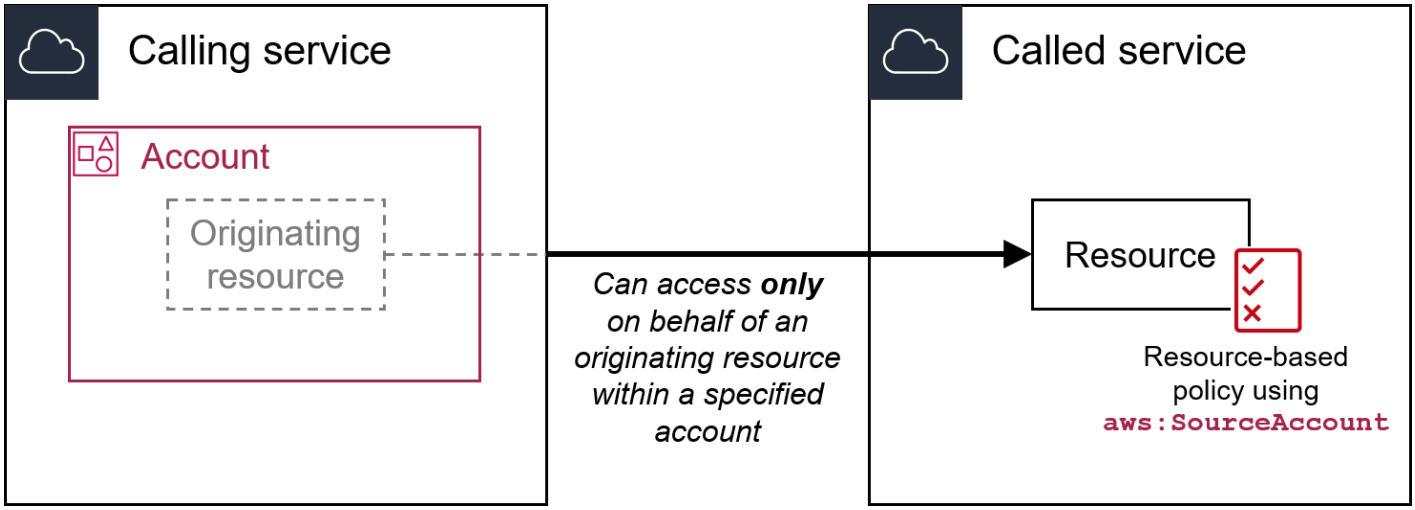
> **Note**
>
> 当 Amazon Web Services 服务 代表您调用其它 Amazon Web Services 服务（服务到服务调用）时，会编辑某些特定于网络的授权上下文。如果您的策略将此条件键与 Deny 语句结合使用，则 Amazon Web Services 服务 主体可能会无意中受到屏蔽。要让 Amazon Web Services 服务 在保持安全要求的同时正常运行，请通过添加值为 false 的 aws:PrincipalIsAWSService 条件键将服务主体排除在 Deny 语句之外。例如：
>
> ```
> {
>   "Effect": "Deny",
>   "Action": "s3:*",
>   "Resource": "*",
>   "Condition": {
>     "Bool": {
>       "aws:SecureTransport": "false",
>       "aws:PrincipalIsAWSService": "false"
>     }
>   }
> }
> ```
>
> 当未使用 HTTPS（aws:SecureTransport 为 false）时，此策略拒绝访问 Amazon S3 操作，但仅适用于非 Amazon 服务主体。这可以确保条件限制适用于除 Amazon Web Services 服务 主体之外的所有主体。

aws:SourceAccount

使用该键可将发出服务到服务请求的资源的账户 ID 与您在策略中指定的账户 ID 进行比较，但仅当请求由 Amazon 服务主体发出时才有效。

- 可用性 – 仅当 Amazon 服务主体代表配置触发服务到服务请求的资源直接调用您的资源时，才会将该键包含在请求上下文中。发出调用的服务将原始资源的账户 ID 传递给被调用的服务。

> **Note**
>
> 此键提供了一种统一机制，用于跨 Amazon Web Services 服务 执行跨服务的混淆代理控制。但是，并非所有服务集成都需要使用此全局条件键。有关缓解跨服务的混淆代理风险的服务特定机制的更多信息，请参阅您使用的 Amazon Web Services 服务 的文档。



- 数据类型 – 字符串
- 值类型 - 单值

您可以使用此条件键，帮助确保调用服务仅能够在请求来自特定账户时访问您的资源。例如，您可以附加以下资源控制策略（RCP）来拒绝服务主体针对 Amazon S3 存储桶提出的请求，除非这些请求是由指定账户中的资源触发的。此策略仅对存在 aws:SourceAccount 密钥的服务主体（"Bool": {"aws:PrincipalIsAWSService": "true"}）（"Null": {"aws:SourceAccount": "false"}）进行的请求应用控制，因此不需要使用此键的服务集成和主体的调用不会受到影响。如果 aws:SourceAccount 键存在于请求上下文中，则 Null 条件将评估为 true，从而强制执行 aws:SourceAccount 键。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "RCPEnforceConfusedDeputyProtection",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceAccount": "111122223333"
        },
        "Null": {
          "aws:SourceAccount": "false"
        },
        "Bool": {
          "aws:PrincipalIsAWSService": "true"
        }
      }
    }
  ]
}
```

在主体为 Amazon Web Services 服务 主体的基于资源的策略中，使用键限制授予服务的权限。例如，当 Amazon S3 存储桶被配置为向 Amazon SNS 主题发送通知时，Amazon S3 服务将为所有已配置的事件调用 `sns:Publish` API 操作。在允许 `sns:Publish` 操作的主题策略中，将条件键的值设置为 Amazon S3 桶的账户 ID。

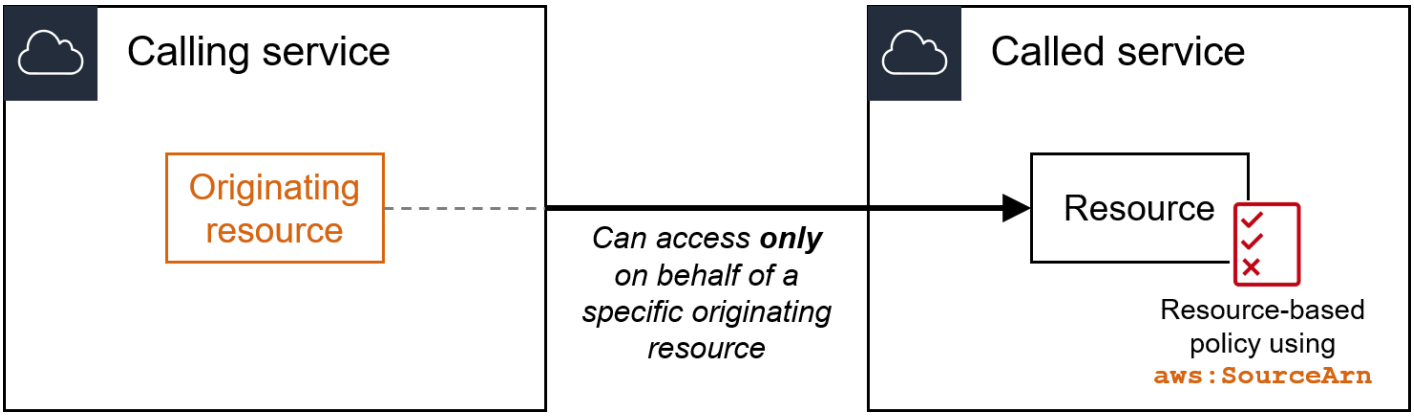
aws:SourceArn

使用该键可将发出服务到服务请求的资源的 Amazon 资源名称 (ARN) 与您在策略中指定的 ARN 进行比较，但仅当请求由 Amazon 服务主体发出时才有效。源的 ARN 包含账户 ID 时，不必将 `aws:SourceAccount` 与 `aws:SourceArn` 结合使用。

此键不适用于发出请求的主体的 ARN。请改用`aws:PrincipalArn`。

- 可用性 – 仅当 Amazon 服务主体代表配置触发服务到服务请求的资源直接调用您的资源时，才会将该键包含在请求上下文中。发出调用的服务将原始资源的 ARN 传递给被调用的服务。

> **Note**
>
> 此键提供了一种统一机制，用于跨 Amazon Web Services 服务 执行跨服务的混淆代理控制。但是，并非所有服务集成都需要使用此全局条件键。有关缓解跨服务的混淆代理风险的服务特定机制的更多信息，请参阅您使用的 Amazon Web Services 服务 的文档。



- 数据类型 – ARN

  Amazon 建议在比较 ARN 时使用 ARN 运算符而不是字符串运算符。

- 值类型 - 单值

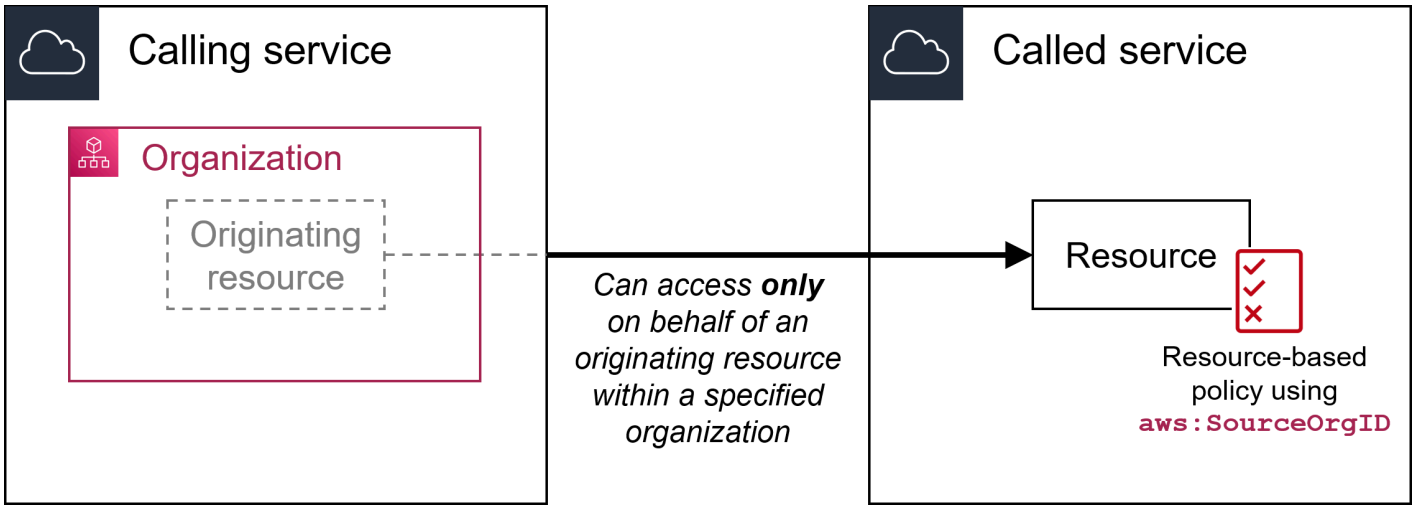
您可以使用此条件键，帮助确保调用服务仅能够在请求来自特定资源时访问您的资源。使用 Amazon Web Services 服务 主体为 `Principal` 的基于资源的策略时，请将此条件键的值设置为您要限制访问的资源的 ARN。例如，当 Amazon S3 存储桶被配置为向 Amazon SNS 主题发送通知时，Amazon S3 服务将为所有已配置的事件调用 `sns:Publish` API 操作。在允许 `sns:Publish` 操作的主题策略中，将条件键的值设置为 Amazon S3 桶的 ARN。有关何时在基于资源的策略中使用此条件键的建议，请参阅您正在使用的 Amazon Web Services 服务 的文档。

aws:SourceOrgID

使用该键可将发出服务到服务请求的资源的组织 ID 与您在策略中指定的组织 ID 进行比较，但仅当请求由 Amazon 服务主体发出时才有效。当您在 Amazon Organizations 中向组织添加和删除账户时，包含 `aws:SourceOrgID` 键的策略将自动包含正确的账户，并且不必手动更新。

- 可用性 – 仅当 Amazon 服务主体代表属于组织成员的账户拥有的资源直接调用您的资源时，才会将该键包含在请求上下文中。发出调用的服务将原始资源的组织 ID 传递给被调用的服务。

> **Note**
>
> 此键提供了一种统一机制，用于跨 Amazon Web Services 服务 执行跨服务的混淆代理控制。但是，并非所有服务集成都需要使用此全局条件键。有关缓解跨服务的混淆代理风险的服务特定机制的更多信息，请参阅您使用的 Amazon Web Services 服务 的文档。



- 数据类型 – 字符串
- 值类型 - 单值

您可以使用此条件键，帮助确保调用服务仅能够在请求来自特定组织时访问您的资源。例如，您可以附加以下资源控制策略（RCP）来拒绝服务主体针对 Amazon S3 存储桶提出的请求，除非这些请求是由指定的 Amazon 组织中的资源触发的。此策略仅对存在 `aws:SourceAccount` 密钥（`"Null": {"aws:SourceAccount": "false"}`）的服务主体（`"Bool": {"aws:PrincipalIsAWSService": "true"}`）进行的请求应用控制，因此不需要使用此键的服务集成和主体的调用不会受到影响。如果 `aws:SourceAccount` 键存在于请求上下文中，则 `Null` 条件将评估为 `true`，从而强制执行 `aws:SourceOrgID` 键。我们在 `Null` 条件运算符中使用 `aws:SourceAccount` 代替 `aws:SourceOrgID`，因此，如果请求来自不属于组织的账户，则控制仍然适用。

JSON

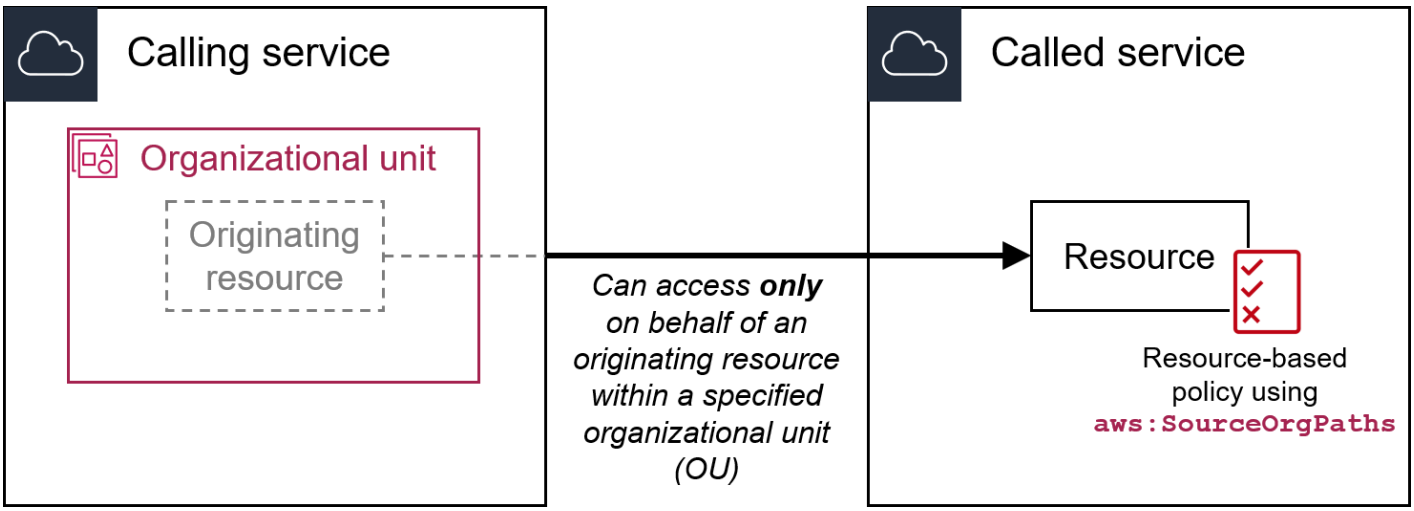
```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Sid": "RCPEnforceConfusedDeputyProtection",
      "Effect": "Deny",
      "Principal": "*",
      "Action": [
        "s3:*"
      ],
      "Resource": "*",
      "Condition": {
        "StringNotEqualsIfExists": {
          "aws:SourceOrgID": "o-xxxxxxxxxx"
        },
        "Null": {
          "aws:SourceAccount": "false"
        },
        "Bool": {
          "aws:PrincipalIsAWSService": "true"
        }
      }
    }
  ]
}
```

aws:SourceOrgPaths

使用该键可将发出服务到服务请求的资源的 Amazon Organizations 路径与您在策略中指定的组织路径进行比较，但仅当请求由 Amazon 服务主体发出时才有效。Amazon Organizations 路径是 Amazon Organizations 实体结构的文本表示形式。有关使用和了解路径的更多信息，请参阅了解 Amazon Organizations 实体路径。

- 可用性 – 仅当 Amazon 服务主体代表属于组织成员的账户拥有的资源直接调用您的资源时，才会将该键包含在请求上下文中。发出调用的服务将原始资源的组织路径传递给被调用的服务。

> **Note**
>
> 此键提供了一种统一机制，用于跨 Amazon Web Services 服务 执行跨服务的混淆代理控制。但是，并非所有服务集成都需要使用此全局条件键。有关缓解跨服务的混淆代理风险的服务特定机制的更多信息，请参阅您使用的 Amazon Web Services 服务 的文档。



- 数据类型 – 字符串（列表）
- 值类型— 多值

使用此条件键，帮助确保调用服务仅能够在请求来自 Amazon Organizations 中特定组织单元（OU）时访问您的资源。

与 aws:SourceOrgID 类似的是，为了帮助防止对不需要使用此键的服务集成造成影响，请结合使用 Null 条件运算符和 aws:SourceAccount 条件键，这样，如果请求来自不属于组织的账户，则该控制仍然适用。

```
{
      "Condition": {
        "ForAllValues:StringNotLikeIfExists": {
            "aws:SourceOrgPaths": "o-a1b2c3d4e5/r-ab12/ou-ab12-11111111/ou-
ab12-22222222/"
        },
        "Null": {
          "aws:SourceAccount": "false"
        },
```

```
        "Bool": {
          "aws:PrincipalIsAWSService": "true"
        }
      }
}
```

aws:SourceOrgPaths 是一个多值条件键。多值键在请求上下文中可以有多个值。对于此键，您必须将 ForAnyValue 或者 ForAllValues 集合运算符与字符串条件运算符搭配使用。有关多值条件键的更多信息，请参阅多值上下文键的集合运算符。

aws:UserAgent

使用此键可将请求者的客户端应用程序与您在策略中指定的应用程序进行比较。

- Availability（可用性）- 此键始终包含在请求上下文中。
- 数据类型 – 字符串
- 值类型— 单值

> **Warning**
>
> 应谨慎使用此键。由于 aws:UserAgent 值由发起人在 HTTP 标头中提供，因此未经授权方可以修改或自定义浏览器以提供他们选择的任何 aws:UserAgent 值。因此，aws:UserAgent 不应用于阻止未经授权方直接发出 Amazon 请求。您可以使用它来仅允许特定客户端应用程序，并且仅在对策略进行测试之后。

aws:IsMcpServiceAction

使用此键可验证正在授权的操作是否是 MCP 服务操作。此键不涉及 MCP 服务对其他 Amazon 服务执行的操作。

- 可用性：此键包含在请求上下文中，仅当 MCP 服务授权 MCP 服务操作时设置为 True。
- 数据类型 – 布尔值
- 值类型 - 单值

其他跨服务条件键

Amazon STS 支持将基于 SAML 的联合身份验证条件键和跨服务条件键用于 OIDC 联合身份验证。当使用 OIDC 或 SAML 联合的用户在其他服务中执行 Amazon 操作时，这些键可用。

IAM 和 Amazon STS 条件上下文密钥

您可以在 JSON 策略中使用 `Condition` 元素来测试所有 Amazon 请求的请求上下文中所包含键的值。这些键提供有关请求本身或请求所引用资源的信息。在允许用户请求的操作之前，您可以检查这些键是否具有指定值。这样，您就可以精细控制 JSON 策略语句何时与传入的请求匹配或不匹配。有关如何在 JSON 策略中使用 `Condition` 元素的信息，请参阅IAM JSON 策略元素：Condition。

本主题介绍了由 IAM 服务定义和提供的键（带 `iam:` 前缀）以及由 Amazon Security Token Service (Amazon STS) 服务定义和提供的键（带 `sts:` 前缀）。其他几个 Amazon 服务也提供与该服务定义的操作和资源相关的服务特定键。有关更多信息，请参阅 Amazon 服务的操作、资源和条件键。支持条件键的服务的文档通常包含附加信息。例如，有关可在 Amazon S3 资源的策略中使用的密钥信息，请参阅 Amazon Simple Storage Service 用户指南中的 Amazon S3 策略密钥。

主题

- IAM 的可用键
- Amazon OIDC 联合身份验证的可用键
- 基于 SAML 的 Amazon STS 联合身份验证的可用键
- 基于 SAML 的跨服务 Amazon STS 联合身份验证上下文键
- Amazon STS 的可用键

IAM 的可用键

可以在控制对 IAM 资源的访问的策略中使用以下条件键：

iam:AssociatedResourceArn

与 ARN 运算符结合使用。

指定此角色在目标服务上关联的资源的 ARN。资源通常属于主体将角色传递到的服务。有时，资源可能属于第三个服务。例如，您可以将角色传递给他们在 Amazon EC2 实例上使用的 Amazon EC2 Auto Scaling。在这种情况下，条件将匹配 Amazon EC2 实例的 ARN。

此条件键仅适用于策略中的 PassRole 操作。无法使用它来限制任何其他操作。

> **Important**
>
> 当在策略中使用 `iam:AssociatedResourceArn` 条件来限制 PassRole 操作时，如果该策略旨在定义 AddRoleToInstanceProfile 操作的访问权限，则需要特别考虑。在这种情况下，您无法在 EC2 实例 ARN 中指定区域或实例 ID。ARN 值必须为 `arn:aws:ec2:*:CallerAccountId:instance/*`。使用任何其他 ARN 值都可能会导致意外的评估结果。

在基于身份的策略中使用此条件键可允许实体传递角色，但前提是角色与指定资源相关联。例如，您可以允许 IAM 用户或角色将任何角色传递给 Amazon EC2 服务，以便与 Amazon Web Services 账户 中的实例一起使用。不允许将 IAM 用户或角色传递给其他服务。

```
{
    "Effect": "Allow",
    "Action": "iam:PassRole",
    "Resource": "*",
    "Condition": {
        "StringEquals": {
            "iam:PassedToService": "ec2.amazonaws.com"
        },
        "ArnLike": {
            "iam:AssociatedResourceARN": [
                "arn:aws:ec2:*:111122223333:instance/*"
            ]
        }
    }
}
```

> **Note**
>
> 支持 iam:PassedToService 的 Amazon 服务也支持此条件键。

iam:AWSServiceName

与字符串运算符结合使用。

指定此角色将附加到的 Amazon 服务。

`CreateServiceLinkedRole` API 操作支持此条件键。

> **Tip**
>
> 有关哪些服务支持使用服务相关角色的信息，请参阅 使用 IAM 的 Amazon 服务 并查找其在服务相关角色列中为是的服务。请选择是与查看该服务的服务关联角色文档的链接。

在此示例中，如果服务名称为 access-analyzer.amazonaws.com，您将允许实体使用 `CreateServiceLinkedRole` API 操作创建与服务关联的角色。

JSON

```
{
   "Version":"2012-10-17",
   "Statement": [{
       "Effect": "Allow",
       "Action": "iam:CreateServiceLinkedRole",
       "Resource": "*",
       "Condition": {
         "StringLike": {
           "iam:AWSServiceName": "access-analyzer.amazonaws.com"
         }
       }
     }]
 }
```

iam:FIDO-certification

与字符串运算符结合使用。

在注册 FIDO 安全密钥时，检查 MFA 设备 FIDO 认证级别。从 FIDO Alliance Metadata Service（MDS）获取设备认证。如果您的 FIDO 安全密钥的认证状态或级别发生变化，则除非设备取消注册，然后重新注册，以获取更新的认证信息，否则不会对其进行更新。

L1、L1plus、L2、L2plus、L3、L3plus 的可能值

在本例中，您注册了安全密钥，并为设备检索 FIDO 1 级 + 认证。

JSON

```
{
      "Version":"2012-10-17",
```

```
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-certification": "L1plus"
                }
            }
        }
    ]

}
```

iam:FIDO-FIPS-140-2-certification

与字符串运算符结合使用。

在注册 FIDO 安全密钥时，检查 MFA 设备 FIPS-140-2 验证认证级别。从 FIDO Alliance Metadata Service（MDS）获取设备认证。如果您的 FIDO 安全密钥的认证状态或级别发生变化，则除非设备取消注册，然后重新注册，以获取更新的认证信息，否则不会对其进行更新。

L1、L2、L3、L4 的可能值

在本例中，您注册了安全密钥，并为设备检索 FIPS-140-2 2 级认证。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
            "Effect": "Allow",
```

```
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-2-certification": "L2"
                }
            }
        }
    ]

}
```

iam:FIDO-FIPS-140-3-certification

与字符串运算符结合使用。

在注册 FIDO 安全密钥时，检查 MFA 设备 FIPS-140-3 验证认证级别。从 FIDO Alliance Metadata Service（MDS）获取设备认证。如果您的 FIDO 安全密钥的认证状态或级别发生变化，则除非设备取消注册，然后重新注册，以获取更新的认证信息，否则不会对其进行更新。

L1、L2、L3、L4 的可能值

在本例中，您注册了安全密钥，并为设备检索 FIPS-140-3 3 级认证。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
```

```
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L3"
                }
            }
        }
    ]

}
```

iam:OrganizationsPolicyId

与字符串运算符结合使用。

检查具有指定 Amazon Organizations ID 的策略是否与请求中使用的策略匹配。要查看使用此条件键的示例 IAM 策略，请参阅。IAM：查看 Amazon Organizations 策略的服务上次访问信息

iam:PassedToService

与字符串运算符结合使用。

指定可将角色传递到的服务的服务主体。此条件键仅适用于策略中的 PassRole 操作。无法使用它来限制任何其他操作。

在策略中使用此条件键时，请使用服务主体指定服务。服务主体是可在策略的 `Principal` 元素中指定的服务的名称。常见格式为：`SERVICE_NAME_URL.amazonaws.com`。

您可以使用 `iam:PassedToService` 限制您的用户，使其只能将角色传递到特定服务。例如，用户可能会创建一个服务角色，该角色信任 CloudWatch 代表用户将日志数据写入到 Amazon S3 存储桶。之后，用户必须将一个权限策略和一个信任策略附加到新的服务角色。在此情况下，信任策略必须指定 `cloudwatch.amazonaws.com` 元素中的 `Principal`。要查看允许用户将角色传递给 CloudWatch 的策略，请参阅 IAM：将 IAM 角色传递给特定 Amazon 服务。

通过使用此条件键，您可以确保用户仅为您指定的服务创建服务角色。例如，如果具有之前的策略的用户尝试为 Amazon EC2 创建服务角色，操作将失败。失败的原因是因为用户无权将角色传递到 Amazon EC2。

有时，您会将角色传递给一个服务，随后将角色传递给另一个服务。`iam:PassedToService` 仅包含担任角色的最终服务，而不包括传递角色的中间服务。

> **Note**
>
> 某些服务不支持此条件键。

iam:PermissionsBoundary

与 ARN 运算符结合使用。

检查指定的策略附加为 IAM 主体资源上的权限边界。有关更多信息，请参阅 IAM 实体的权限边界

iam:PolicyARN

与 ARN 运算符结合使用。

检查涉及托管策略的请求中的托管策略的 Amazon 资源名称（ARN）。有关更多信息，请参阅 控制对策略的访问。

iam:RegisterSecurityKey

与字符串运算符结合使用。

检查 MFA 设备支持的当前状态。

`Create` 或 `Activate` 的可能值。

在本例中，您注册了安全密钥，并为设备检索 FIPS-140-3 1 级认证。

JSON

```
{
      "Version":"2012-10-17",
      "Statement": [{
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
```

```
                    "iam:RegisterSecurityKey" : "Create"
                }
            }
        },
        {
            "Effect": "Allow",
            "Action": "iam:EnableMFADevice",
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "iam:RegisterSecurityKey" : "Activate",
                    "iam:FIDO-FIPS-140-3-certification": "L1"
                }
            }
        }
    ]

}
```

iam:ResourceTag/***key-name***

与字符串运算符结合使用。

检查附加到身份资源（用户或角色）的标签是否与指定的键名称和键值匹配。

> **Note**
>
> IAM 和 Amazon STS 同时支持 `iam:ResourceTag` IAM 条件键和 `aws:ResourceTag` 全局条件键。

您可采用键值对的形式向 IAM 资源添加自定义属性。有关 IAM 资源的标签的更多信息，请参阅 the section called “IAM 资源的标签”。您可以使用 `ResourceTag` 控制对 Amazon 资源（包括 IAM 资源）的访问。但是，由于 IAM 不支持组的标签，因此您不能使用标签来控制对组的访问。

此示例说明如何创建基于身份的策略以允许删除具有 **`status=terminated`** 标签的用户。要使用此策略，请将示例策略中的*******替换为您自己的信息。然后，按照创建策略或编辑策略中的说明操作。

JSON

```
{
```

```
    "Version":"2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": "iam:DeleteUser",
        "Resource": "*",
        "Condition": {"StringEquals": {"iam:ResourceTag/status":
 "terminated"}}
    }]
}
```

iam:ServiceSpecificCredentialAgeDays

与数字运算符结合使用。

此条件键根据过期设置限制特定服务凭证的创建。通过此键，您可以控制可创建的特定服务凭证的最大期限（以天为单位）。

有效天数范围为 1 到 36600 天（最短 1 天，最长 36600 天）。

CreateServiceSpecificCredential API 操作支持此条件键。

在此示例中，只有在 90 天内过期时，才允许用户为 Amazon Bedrock 服务创建特定服务凭证。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceSpecificCredential",
            "Resource": "arn:aws:iam::111122223333:user/username",
            "Condition": {
                "StringEquals": {
                    "iam:ServiceSpecificCredentialServiceName":
 "bedrock.amazonaws.com"
                },
                "NumericLessThanEquals": {
                    "iam:ServiceSpecificCredentialAgeDays": "90"
                }
            }
        }
    ]
```

```
}
```

iam:ServiceSpecificCredentialServiceName

与字符串运算符结合使用。

指定在管理特定服务凭证时可以使用哪些 Amazon 服务。通过此条件键，您可以限制在管理特定服务凭证时允许使用哪些 Amazon 服务。

以下 API 操作支持此条件键：

- CreateServiceSpecificCredential
- DeleteServiceSpecificCredential
- ResetServiceSpecificCredential
- UpdateServiceSpecificCredential

以下服务支持特定服务凭证及其精确值格式：

- bedrock.amazonaws.com
- cassandra.amazonaws.com
- codecommit.amazonaws.com

在此示例中，允许用户仅使用 CreateServiceSpecificCredential API 操作为 Amazon Bedrock 服务创建特定服务凭证。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateServiceSpecificCredential",
            "Resource": "arn:aws:iam::111122223333:user/username",
            "Condition": {
                "StringEquals": {
                    "iam:ServiceSpecificCredentialServiceName":
 "bedrock.amazonaws.com"
                }
            }
        }
    ]
```

```
}
```

iam:DelegationDuration

与数字运算符结合使用。

根据在委派请求中请求临时访问的持续时间筛选访问权限。

产品提供商可以使用此条件键来控制其允许向客户发送的委派请求的最大持续时间。持续时间以秒为单位指定，其决定了客户发放交换令牌后临时凭证保持有效的时长。这有助于产品提供商根据其使用案例强制执行有关访问持续时间限制的内部策略。

`CreateDelegationRequest` API 操作支持此条件键。

在此示例中，仅当请求的持续时间为 7200 秒（2 小时）或更短时，才允许创建委派请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateDelegationRequest",
            "Resource": "*",
            "Condition": {
                "NumericLessThanEquals": {
                    "iam:DelegationDuration": "7200"
                }
            }
        }
    ]
}
```

iam:NotificationChannel

与 ARN 运算符结合使用。

根据为接收委派请求通知指定的 Amazon SNS 主题 ARN 筛选访问权限。

产品提供商可以使用此条件键来限制哪些 SNS 主题可用于 CreateDelegationRequest API 调用中的委派请求通知。产品提供商必须指定 SNS 主题才能接收状态变更通知和交换令牌。这可确保通知仅发送到产品提供商组织内经批准的渠道。

`CreateDelegationRequest` API 操作支持此条件键。

在此示例中，仅当委派请求使用特定 SNS 主题进行通知时，才允许创建委派请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateDelegationRequest",
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
                    "iam:NotificationChannel": "arn:aws:sns:us-
east-1:123456789012:delegation-notifications"
                }
            }
        }
    ]
}
```

iam:TemplateArn

与 ARN 运算符结合使用。

根据用于在委派请求中定义权限的策略模板 ARN 筛选访问权限。

产品提供商可以使用此条件键来控制哪些策略模板能够在 CreateDelegationRequest API 调用中使用。策略模板定义产品提供商在客户账户中请求的临时权限。这允许产品提供商限制在创建委派请求时可以使用哪些注册的策略模板。

`CreateDelegationRequest` API 操作支持此条件键。

在此示例中，仅当委派请求使用来自特定合作伙伴域的策略模板时，才允许创建委派请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "iam:CreateDelegationRequest",
            "Resource": "*",
            "Condition": {
                "ArnLike": {
                    "iam:TemplateArn": "arn:aws:iam:::delegation-template/partner_*"
```

```
                }
            }
        }
    ]
}
```

iam:DelegationRequestOwner

与 ARN 运算符结合使用。

根据拥有委派请求的 Amazon 身份或主体筛选访问权限。

客户可以使用此条件键来控制哪些人员可以根据所有权对委派请求执行操作。委派请求的所有者是发起或接收委派请求的客户账户中的 Amazon 身份或主体。

以下 API 操作支持此条件键：

- `GetDelegationRequest`
- `AcceptDelegationRequest`
- `RejectDelegationRequest`
- `SendDelegatedToken`
- `ListDelegationRequests`
- `UpdateDelegationRequest`

在此示例中，仅允许用户管理其拥有的委派请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "iam:GetDelegationRequest",
                "iam:AcceptDelegationRequest",
                "iam:RejectDelegationRequest",
                "iam:SendDelegatedToken",
                "iam:UpdateDelegationRequest",
                "iam:ListDelegationRequests"
            ],
            "Resource": "*",
            "Condition": {
                "ArnEquals": {
```

```
                    "iam:DelegationRequestOwner": "${aws:PrincipalArn}"
                }
            }
        }
    ]
}
```

Amazon OIDC 联合身份验证的可用键

您可以使用 OIDC 联合身份验证，为已通过 OpenID Connect 兼容的身份提供者 (IdP) 向您的 Amazon 账户中的 IAM OpenID Connect（OIDC）身份提供者进行身份验证的用户提供临时安全凭证。此类提供者的示例包括 GitHub、Amazon Cognito、Login with Amazon 和 Google。可以使用您自己的 IdP 中的身份令牌和访问令牌，也可以使用授给 Amazon Elastic Kubernetes 服务工作负载的服务账户令牌。

您可以使用 Amazon OIDC 条件上下文键编写策略，将联合主体的访问权限限制为与特定提供者、应用或用户关联的资源。这些键通常在角色的信任策略中使用。使用 OIDC 提供者的名称 (`token.actions.githubusercontent.com`) 和声明 (`:aud`) 定义条件键：**`token.actions.githubusercontent.com:aud`**。

某些 OIDC 联合身份验证条件键可以在角色会话中用于授权资源访问权限。如果“在会话中可用”列中的值为“是”，则可以在策略中使用这些条件键来定义其他 Amazon 服务中允许用户访问的内容。当声明在会话中不可用时，OIDC 条件上下文键只能在角色信任策略中用于初始 AssumeRoleWithWebIdentity 身份验证。

选择您的 IdP，查看 IdP 中的声明如何映射到 Amazon 中的 IAM 条件上下文键。有关 GitHub 和 Google 密钥的更多信息，可在默认选项卡下找到。

Default

默认列出了标准 OIDC 声明以及它们如何映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。如果选项卡选项中未列出您的 IdP，则使用此映射。

GitHub Actions 工作流和 Google 是在 OIDC JWT ID 令牌中使用默认实现的 IDP 的一些示例。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
amr	amr	是

Amazon STS 条件键	IdP JWT 声明	在会话中可用
aud	azp 如果未为 azp 设置任何值，则 aud 条件键将映射到 aud 声明。	是
电子邮件	电子邮件	否
oaud	aud	否
sub	sub	是

有关将 OIDC 条件上下文键与 GitHub 结合使用的更多信息，请参阅 为 GitHub OIDC 身份提供程序配置角色。有关 Google aud 和 azp 字段的更多信息，请参阅 Google Identity Platform OpenID Connect 指南。

amr

与字符串运算符结合使用。该键有多个值，这意味着您要在使用条件集合运算符的策略中对它进行测试。

示例：`token.actions.githubusercontent.com:amr`

“身份验证方法引用”包括有关用户的登录信息。该键可包含以下值：

- 如果用户未经过身份验证，则该键仅包含 `unauthenticated`。
- 如果用户已通过身份验证，则该键包含值 `authenticated` 以及调用 (`accounts.google.com`) 中使用的登录提供者的名称。

aud

与字符串运算符结合使用。

示例：

- `accounts.google.com:aud`
- `token.actions.githubusercontent.com:aud`

使用 aud 条件键验证受众是否与您在策略中指定的内容匹配。对于同一身份提供者，可以将 aud 键与 sub 键结合使用。

此条件键是从以下令牌字段设置的：

- 应用程序的 OAuth 2.0 Google 客户端 ID 的 aud（如果未设置 azp 字段）。如果设置了 azp 字段，则 aud 字段将与 accounts.google.com:oaud 条件键匹配。
- azp（如果设置了 azp 字段）。对于 Web 应用程序和 Android 应用程序具有不同的 OAuth 2.0 Google 客户端 ID 但共享相同的 Google API 项目的混合应用程序，可能会出现这种情况。

在使用 accounts.google.com:aud 条件键编写策略时，您必须了解该应用程序是否为设置 azp 字段的混合应用程序。

azp 字段未设置

以下示例策略适用于未设置 azp 字段的非混合应用程序。在此情况下，Google ID 令牌 aud 字段值将与 accounts.google.com:aud 和 accounts.google.com:oaud 条件键值匹配。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "accounts.google.com:aud": "aud-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

azp 字段已设置

以下示例策略适用于设置 azp 字段的混合应用程序。在此情况下，Google ID 令牌 aud 字段值仅与 accounts.google.com:oaud 条件键值匹配。azp 字段值与 accounts.google.com:aud 条件键值匹配。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {"Federated": "accounts.google.com"},
            "Action": "sts:AssumeRoleWithWebIdentity",
            "Condition": {
                "StringEquals": {
                    "accounts.google.com:aud": "azp-value",
                    "accounts.google.com:oaud": "aud-value",
                    "accounts.google.com:sub": "sub-value"
                }
            }
        }
    ]
}
```

email

与字符串运算符结合使用。

示例：`accounts.google.com:email`

此条件键验证用户的电子邮件地址。此声明的值可能不是此账户所独有的，并且可能会随着时间的推移而发生变化，因此您不应使用此值作为主要标识符来验证用户记录。

oaud

与字符串运算符结合使用。

示例：`accounts.google.com:oaud`

此键指定此 ID 令牌用于的其他受众 (aud)。它必须是应用程序的 OAuth 2.0 客户端 ID 之一。

sub

与字符串运算符结合使用。

示例：

- `accounts.google.com:sub`

- token.actions.githubusercontent.com:sub

使用这些键验证使用者是否与您在策略中指定的内容匹配。对于同一身份提供者，可以将 sub 键与 aud 键结合使用。

在以下角色信任策略中，sub 条件键将角色限制为名为 demo 的 GitHub 分支。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::111122223333:oidc-provider/
token.actions.githubusercontent.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
          "token.actions.githubusercontent.com:sub": "repo:org-name/repo-
name:ref:refs/heads/demo"
        }
      }
    }
  ]
}
```

Amazon Cognito

此选项卡说明了 Amazon Cognito 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

对于 Amazon Cognito 使用的角色，键是使用 `cognito-identity.amazonaws.com` 和声明定义的。

有关身份池声明映射的更多信息，请参阅《Amazon Cognito 开发人员指南》中的默认提供商映射。有关用户池声明映射的更多信息，请参阅《Amazon Cognito 开发人员指南》中的使用 ID 令牌。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
amr	amr	是
aud	aud	是
oaud	aud	否
sub	sub	是

amr

与字符串运算符结合使用。该键有多个值，这意味着您要在使用条件集合运算符的策略中对它进行测试。

示例 – `cognito-identity.amazonaws.com:amr`

“身份验证方法引用”包括有关用户的登录信息。该键可包含以下值：

- 如果用户未经过身份验证，则该键仅包含 `unauthenticated`。
- 如果用户已通过身份验证，则该键包含值 `authenticated` 以及调用 (`cognito-identity.amazonaws.com`) 中使用的登录提供者的名称。

例如，某 Amazon Cognito 角色信任策略中的以下条件测试用户是否未经身份验证：

```
"Condition": {
  "StringEquals":
    { "cognito-identity.amazonaws.com:aud": "us-east-2:identity-pool-id" },
  "ForAnyValue:StringLike":
    { "cognito-identity.amazonaws.com:amr": "unauthenticated" }
}
```

aud

与字符串运算符结合使用。

示例 – `cognito-identity.amazonaws.com:aud`

对用户进行身份验证的用户池应用程序客户端。Amazon Cognito 在访问令牌 `client_id` 声明中呈现相同的值。

oaud

与字符串运算符结合使用。

示例 – `cognito-identity.amazonaws.com:oaud`

对用户进行身份验证的用户池应用程序客户端。Amazon Cognito 在访问令牌 `client_id` 声明中呈现相同的值。

sub

与字符串运算符结合使用。

示例 – `cognito-identity.amazonaws.com:sub`

经过身份验证的用户的唯一标识符（UUID）或主题。用户名在您的用户池中可能不是唯一的。此子声明是识别给定用户的最佳方法。对于同一身份提供者，可以将 `sub` 键与 `aud` 键结合使用。

```
"Condition": {
         "StringEquals": {
            "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
            "cognito-identity.amazonaws.com:sub": [
               "us-east-1:12345678-1234-1234-1234-123456790ab",
               "us-east-1:98765432-1234-1234-1243-123456790ab"
            ]
         }
      }
```

Login with Amazon

此选项卡说明了 Login with Amazon 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
app_id	应用程序 ID	是
sub	用户 ID	是

Amazon STS 条件键	IdP JWT 声明	在会话中可用
user_id	用户 ID	是

app_id

与字符串运算符结合使用。

示例 – `www.amazon.com:app_id`

此键指定与其他身份提供者使用的 `aud` 字段匹配的受众上下文。

sub

与字符串运算符结合使用。

示例 – `www.amazon.com:sub`

此键验证用户 ID 是否与您在策略中指定的内容匹配。对于同一身份提供者，可以将 `sub` 键与 `aud` 键结合使用。

user_id

与字符串运算符结合使用。

示例 – `www.amazon.com:user_id`

此键指定与其他身份提供者使用的 `aud` 字段匹配的受众上下文。对于同一身份提供者，可以将 `user_id` 键与 `id` 键结合使用。

Facebook

此选项卡说明了 Facebook 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
app_id	应用程序 ID	是
id	id	是

app_id

与字符串运算符结合使用。

示例 – `graph.facebook.com:app_id`

此键验证受众上下文是否与其他身份提供者使用的 aud 字段匹配。

id

与字符串运算符结合使用。

示例 – `graph.facebook.com:id`

此键验证应用程序（或站点）ID 是否与您在策略中指定的内容匹配。

GitHub

此选项卡说明了 GitHub Actions 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
演员	演员	否
actor_id	actor_id	否
job_workflow_ref	job_workflow_ref	否
repository	repository	否
repository_id	repository_id	否
workflow	workflow	否
ref	ref	否
环境	环境	否
enterprise_id	enterprise_id	否

演员

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:actor`

此密钥标识启动工作流程运行的个人账户。使用此密钥限制对特定角色的访问权限

actor_id

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:actor_id`

此密钥验证启动工作流程运行的个人账户的 ID。角色 ID 由 GitHub 生成，并且是不可变的。

job_workflow_ref

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:job_workflow_ref`

对于使用可重用工作流程的作业，此密钥包含可重用工作流程的参考路径。使用此密钥限制对特定工作流程的访问权限，并确保只有经过批准的工作流程才能担任角色。

repository

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:repository`

此密钥标识运行工作流程的存储库。使用此密钥限制对特定 GitHub 存储库的访问权限。

repository_id

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:repository_id`

此密钥验证运行工作流程的存储库的 ID。存储库 ID 是不可变的，即使存储库被重命名也不会更改。

workflow

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:workflow`

此密钥包含工作流程的名称。使用此密钥限制对存储库中特定工作流程的访问权限。

ref

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:ref`

此密钥标识触发工作流程运行的 git ref（分支或标签）。使用此密钥根据特定分支限制访问权限，例如仅允许 `main` 或 `production` 分支。

环境

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:environment`

此密钥包含作业使用的环境的名称。使用此密钥实现基于环境的访问控制，例如开发环境、暂存环境和生产环境的单独权限。

> **Note**
>
> 如果您的信任策略中包含环境声明，则必须在 GitHub 工作流程中配置和提供环境。

enterprise_id

与字符串运算符结合使用。

示例 – `token.actions.githubusercontent.com:enterprise_id`

此密钥验证企业的 ID，该企业包含运行工作流程的存储库。使用此密钥确保访问权限仅限于您的 GitHub Enterprise 组织内的存储库。

以下示例信任策略使用 GitHub OIDC 令牌中的自定义声明来限制对角色的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
```

```
            "Federated": "arn:aws:iam::AWS_ACCOUNT_ID:oidc-provider/
token.actions.githubusercontent.com"
         },
         "Action": "sts:AssumeRoleWithWebIdentity",
         "Condition": {
            "StringLike": {
                "token.actions.githubusercontent.com:aud": "sts.amazonaws.com",
                "token.actions.githubusercontent.com:job_workflow_ref": "octo-org/
octo-automation/.github/workflows/oidc.yml@refs/heads/main",
                "token.actions.githubusercontent.com:repository": "octo-org/octo-
repo",
                "token.actions.githubusercontent.com:actor": "octocat",
                "token.actions.githubusercontent.com:ref": "refs/heads/main",
                "token.actions.githubusercontent.com:enterprise_id": "345"
               }
           }
        }
    ]
}
```

Google

此选项卡说明了 Google 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
google/organization_number	google:organization_number	否

google/organization_number

与数字运算符结合使用。

示例 – `accounts.google.com:google/organization_number`

此密钥验证令牌是否代表属于特定 Google Cloud 或 Google Workspace 组织的 Google 身份。使用此密钥限制对来自特定组织的用户的访问权限，确保只有您所在组织的身份才能担任该角色。

以下示例信任策略使用 `google/organization_number` 声明来限制对角色的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Federated": "accounts.google.com"
         },
         "Action": "sts:AssumeRoleWithWebIdentity",
         "Condition": {
            "NumericEquals": {
                "accounts.google.com:google/organization_number": "123456"
               }
           }
        }
    ]
}
```

CircleCI

此选项卡说明了 CircleCI 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
oidc.circleci.com/project-id	oidc.circleci.com/project-id	否

oidc.circleci.com/project-id

与字符串运算符结合使用。

示例 – `circleci-issuer-url:oidc.circleci.com/project-id`

此密钥标识运行作业的 CircleCI 项目。其值是一个包含 UUID 的字符串，该 UUID 唯一标识 CircleCI 项目。使用此密钥限制对特定 CircleCI 项目的访问权限。

以下示例信任策略使用 `oidc.circleci.com/project-id` 声明来限制对角色的访问权限。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/oidc.circleci.com/
org/12345"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "oidc.circleci.com/org/12345:aud": "sts.amazonaws.com",
          "oidc.circleci.com/org/12345:oidc.circleci.com/project-id": "76543210-
ba98-fedc-3210-edcba0987654"
        }
      }
    }
  ]
}
```

Oracle Cloud Infrastructure (OCI)

此选项卡说明了 Oracle Cloud Infrastructure 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
rpst_id	rpst_id	否

rpst_id

与字符串运算符结合使用。

示例 – `oci-issuer-url:rpst_id`

此密钥唯一标识 OCI 中的资源主体。使用此密钥限制对特定 OCI 资源主体的访问权限。rpst_id（资源主体会话令牌 ID）为基于 OCI 资源的身份验证提供稳定的标识符。

以下示例信任策略使用 `rpst_id` 声明来限制对角色的访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/idcs-
abc123ef5678901234abcd.identity.oraclecloud.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "idcs-abc123ef5678901234abcd.identity.oraclecloud.com:aud":
 "sts.amazonaws.com",
          "idcs-abc123ef5678901234abcd.identity.oraclecloud.com:rpst_id": "your-
rpst-id"
        }
      }
    }
  ]
}
```

GitLab.com

此选项卡说明了 GitLab.com CI/CD 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

> **Note**
>
> 以下自定义条件键仅在将 GitLab.com 多租户 SaaS 实例用作 OIDC 身份提供者时受支持。您不能将这些条件键与 GitLab Dedicated 或 GitLab Self-Managed 实例一起使用。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
namespace_id	namespace_id	否
project_id	project_id	否
user_id	user_id	否

Amazon STS 条件键	IdP JWT 声明	在会话中可用
user_login	user_login	否
user_email	user_email	否
user_access_level	user_access_level	否
ref_protected	ref_protected	否
pipeline_source	pipeline_source	否

namespace_id

与字符串运算符结合使用。

示例 – `gitlab.com:namespace_id`

此键标识运行 CI/CD 作业的项目的命名空间（组）ID。命名空间 ID 是一个稳定、唯一的标识符，即使组被重命名也不会改变。

project_id

与字符串运算符结合使用。

示例 – `gitlab.com:project_id`

此键标识运行 CI/CD 作业的项目的 ID。项目 ID 是一个稳定、唯一的标识符，即使项目被重命名或移动也不会改变。

user_id

与字符串运算符结合使用。

示例 – `gitlab.com:user_id`

此键标识执行 CI/CD 作业的用户的 ID。用户 ID 是一个稳定、唯一的标识符。

user_login

与字符串运算符结合使用。

示例 – `gitlab.com:user_login`

此键标识执行 CI/CD 作业的用户的用户名。用户名可以更改；如需稳定的标识符，请改用 `user_id`。

user_email

与字符串运算符结合使用。

示例 – `gitlab.com:user_email`

此键标识执行 CI/CD 作业的用户的电子邮件地址。此声明的值可能随时间变化；如需稳定的标识符，请改用 `user_id`。

user_access_level

与字符串运算符结合使用。

示例 – `gitlab.com:user_access_level`

此键标识用户在项目中的访问级别（例如`maintainer`、`developer`、`owner`）。

ref_protected

与字符串运算符结合使用。

示例 – `gitlab.com:ref_protected`

此键指示触发作业的 Git 引用是否受保护。如果受保护，值为 `true`，否则为 `false`。

pipeline_source

与字符串运算符结合使用。

示例 – `gitlab.com:pipeline_source`

此键标识触发管道的来源（例如`push`、`web`、`schedule`、`api`、`merge_request_event`）。

Important

在 Gitlab.com SaaS 上，已删除的群组或项目路径以前可以由不同的用户重新创建，这可能会导致策略仅依赖基于路径的声明（例如 `sub`）来授予对非预期身份的访问权限。自 2026 年 6 月 1 日起，GitLab 已经部署了平台端更新，该更新可防止从以前发出过令牌的项目路径生成新的 CI ID 令牌，从而在未来降低了此类风险。作为额外的保护层，Amazon 和

GitLab 在角色信任策略或资源控制策略中包含基于稳定唯一标识符（例如 namespace_id 或 project_id）的条件，以缓解此风险。

以下示例信任策略使用 namespace_id 声明结合 sub 将访问权限限制到特定的 GitLab.com 项目和分支。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::123456789012:oidc-provider/gitlab.com"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "gitlab.com:sub": "project_path:mygroup/
myproject:ref_type:branch:ref:main",
          "gitlab.com:namespace_id": "12345"
        }
      }
    }
  ]
}
```

Buildkite

此选项卡说明了 Buildekite 如何将 OIDC 声明映射到 Amazon 中的 Amazon STS 条件上下文键。您可以使用这些键来控制对角色的访问。为此，请将 Amazon STS 条件键与 IdP JWT 声明列中的值进行比较。

Amazon STS 条件键	IdP JWT 声明	在会话中可用
organization_slug	organization_slug	否
organization_id	organization_id	否
pipeline_slug	pipeline_slug	否

Amazon STS 条件键	IdP JWT 声明	在会话中可用
pipeline_id	pipeline_id	否
cluster_name	cluster_name	否
cluster_id	cluster_id	否
build_branch	build_branch	否

organization_slug

与字符串运算符结合使用。

示例 – `agent.buildkite.com:organization_slug`

此密钥标识运行作业的 Buildkite 组织。使用此密钥限制对特定组织的访问权限。

organization_id

与字符串运算符结合使用。

示例 – `agent.buildkite.com:organization_id`

此密钥验证运行作业的 Buildkite 组织 ID。组织 ID 由 Buildkite 生成，且是不可变的。

pipeline_slug

与字符串运算符结合使用。

示例 – `agent.buildkite.com:pipeline_slug`

此密钥标识运行作业的管道。使用此密钥限制对特定 Buildkite 管道的访问权限。

pipeline_id

与字符串运算符结合使用。

示例 – `agent.buildkite.com:pipeline_id`

此密钥验证运行作业的管道 ID。管道 ID 由 Buildkite 生成，且是不可变的。

cluster_name

与字符串运算符结合使用。

示例 – `agent.buildkite.com:cluster_name`

此密钥标识运行作业的集群。使用此密钥限制对特定 Buildkite 集群的访问权限。

cluster_id

与字符串运算符结合使用。

示例 – `agent.buildkite.com:cluster_id`

此密钥验证运行作业的集群 ID。集群 ID 由 Buildkite 生成，且是不可变的。

build_branch

与字符串运算符结合使用。

示例 – `agent.buildkite.com:build_branch`

此密钥标识触发构建的 git 分支。使用此密钥根据特定分支限制访问权限，例如仅允许 `main` 或 `production` 分支。

以下示例信任策略使用 Buildkite OIDC 令牌中的自定义声明来限制对角色的访问权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
    {
        "Effect": "Allow",
        "Principal": {
            "Federated": "arn:aws:iam::123456789012:oidc-provider/
agent.buildkite.com"
        },
        "Action": "sts:AssumeRoleWithWebIdentity",
        "Condition": {
            "StringEquals": {
                "agent.buildkite.com:aud": "sts.amazonaws.com",
                "agent.buildkite.com:organization_slug": "acme-inc",
                "agent.buildkite.com:pipeline_slug": "super-duper-app",
                "agent.buildkite.com:build_branch": "main"
            }
        }
    }
    ]
}
```

有关 OIDC 联合身份验证的更多信息

- Amazon Cognito 用户指南
- OIDC 联合身份验证

基于 SAML 的 Amazon STS 联合身份验证的可用键

如果您通过 Amazon Security Token Service (Amazon STS) 使用基于 SAML 的联合身份验证，则可以在策略中包含更多条件键。

SAML 角色信任策略

在角色的信任策略中，您可以包括以下键，以帮助您确定发起人是否有权担任角色。除了 `saml:doc`，所有值均源自 SAML 断言。在创建或编辑带条件的策略时，可在 IAM 控制台可视化编辑器中使用列表中的所有项目。标有 `[]` 的项目可以 具有指定类型列表中的值。

saml:aud

与字符串运算符结合使用。

SAML 断言提交到的端点 URL。此键的值来自断言中的 `SAML Recipient` 字段，而不是 `Audience` 字段。

saml:commonName[]

与字符串运算符结合使用。

这是 `commonName` 属性。

saml:cn[]

与字符串运算符结合使用。

这是 `eduOrg` 属性。

saml:doc

与字符串运算符结合使用。

这代表担任角色所用的主体。格式为 *`account-ID`*/*`provider-friendly-name`*，例如 `123456789012/SAMLProviderName`。账户 ID 值指拥有 SAML 提供商的账户。

saml:edupersonaffiliation[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonassurance[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonentitlement[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonnickname[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonorgdn

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonorgunitdn[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonprimaryaffiliation

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonprimaryorgunitdn

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonprincipalname

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersonscopedaffiliation[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:edupersontargetedid[]

与字符串运算符结合使用。

这是 eduPerson 属性。

saml:eduorghomepageuri[]

与字符串运算符结合使用。

这是 eduOrg 属性。

saml:eduorgidentityauthnpolicyuri[]

与字符串运算符结合使用。

这是 eduOrg 属性。

saml:eduorglegalname[]

与字符串运算符结合使用。

这是 eduOrg 属性。

saml:eduorgsuperioruri[]

与字符串运算符结合使用。

这是 eduOrg 属性。

saml:eduorgwhitepagesuri[]

与字符串运算符结合使用。

这是 eduOrg 属性。

saml:givenName[]

与字符串运算符结合使用。

这是 `givenName` 属性。

saml:iss

与字符串运算符结合使用。

发布者，以 URN 表示。

saml:mail[]

与字符串运算符结合使用。

这是 `mail` 属性。

saml:name[]

与字符串运算符结合使用。

这是 `name` 属性。

saml:namequalifier

与字符串运算符结合使用。

基于 SAML 提供商的友好名称的哈希值。该值是以下值按顺序的连接，以“/”字符分隔：

1. `Issuer` 响应值 (`saml:iss`)
2. `AWS` 账户 ID
3. IAM 中 SAML 提供商的友好名称（ARN 的最后一部分）

账户 ID 与 SAML 提供商的易记名称的串联可作为键 `saml:doc` 供 IAM 策略使用。有关更多信息，请参阅 唯一标识基于 SAML 的联合身份验证中的用户。

saml:organizationStatus[]

与字符串运算符结合使用。

这是 `organizationStatus` 属性。

saml:primaryGroupSID[]

与字符串运算符结合使用。

这是 `primaryGroupSID` 属性。

saml:sub

与字符串运算符结合使用。

这是该陈述的主题，其中包含唯一标识组织中某个用户的值 (例如 _cbb88bf52c2510eabe00c1642d4643f41430fe25e3)。

saml:sub_type

与字符串运算符结合使用。

此键的值可以是 persistent、transient 或由 SAML 断言中使用的 Format 和 Subject 元素的完整 NameID URI 构成。persistent 值表示在不同会话之间用户的 saml:sub 值是相同的。如果值为 transient，则用户在每个会话中拥有不同的 saml:sub 值。有关 NameID 元素的 Format 属性的信息，请参阅为身份验证响应配置 SAML 断言。。

saml:surname[]

与字符串运算符结合使用。

这是 surnameuid 属性。

saml:uid[]

与字符串运算符结合使用。

这是 uid 属性。

saml:x500UniqueIdentifier[]

与字符串运算符结合使用。

这是 x500UniqueIdentifier 属性。

有关 eduPerson 和 eduOrg 属性的一般信息，请参阅 REFEDS Wiki 网站。有关 eduPerson 属性的列表，请参阅 eduPerson 对象类规范 (201602)。

列表类型的条件键可以包含多个值。要在策略中创建条件以获取列表值，可以使用集合运算符（ForAllValues、ForAnyValue）。例如，要支持从属关系为“教员”或“职员”（而非“学生”）的用户，可以使用下面这样的条件：

```
"Condition": {
   "ForAllValues:StringLike": {
     "saml:edupersonaffiliation":[ "faculty", "staff"]
   }
```

```
}
```

基于 SAML 的跨服务 Amazon STS 联合身份验证上下文键

一些基于 SAML 的联合身份验证条件键可用于后续请求中，以授权其他服务和 `AssumeRole` 调用中的 Amazon 操作。以下是条件键，当联合主体担任其他角色时，可以在角色信任策略中使用它们，也可以在其他 Amazon 服务的资源策略中使用它们来授权联合主体访问资源。有关使用这些键的更多信息，请参阅关于基于 SAML 2.0 的联合身份验证。

选择条件键以查看描述。

- saml:namequalifier
- saml:sub
- saml:sub_type

> **Note**
>
> 在初始外部身份提供者 (IdP) 身份验证响应后，没有其他基于 SAML 的联合身份验证条件键可供使用。

Amazon STS 的可用键

您可以对使用 Amazon Security Token Service (Amazon STS) 操作担任的角色使用 IAM 角色信任策略中的以下条件键。

saml:sub

与字符串运算符结合使用。

这是该陈述的主题，其中包含唯一标识组织中某个用户的值 (例如 `_cbb88bf52c2510eabe00c1642d4643f41430fe25e3`)。

sts:AWSServiceName

与字符串运算符结合使用。

使用此键指定可以在其中使用持有者令牌的服务。在策略中使用此条件键时，请使用服务主体指定服务。服务主体是可在策略的 `Principal` 元素中指定的服务的名称。例如，`codeartifact.amazonaws.com` 是 Amazon CodeArtifact 服务主体。

Availability（可用性）- 此键存在于获取持有者令牌的请求中。您不能直接调用 Amazon STS 以获取持有者令牌。当您在其他服务中执行某些操作时，服务代表您请求持有者令牌。

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。例如，Amazon CodeArtifact 要求主体使用持有者令牌来执行某些操作。aws `codeartifact get-authorization-token` 命令返回一个持有者令牌。然后，您可以使用持有者令牌来执行 Amazon CodeArtifact 操作。有关持有者令牌的更多信息，请参阅 服务持有者令牌。

您可以使用此条件键以允许主体获取用于特定服务的持有人令牌。

sts:DurationSeconds

与数字运算符结合使用。

使用此键指定主体在从 GetWebIdentityToken API 获取 Amazon Amazon STS 持有者令牌或 JSON Web 令牌时可以使用的持续时间（以秒为单位）。

可用性：此键存在于从 GetWebIdentityToken API 获取持有者令牌或 JSON Web 令牌的请求中。您不能直接调用 Amazon STS 以获取持有者令牌。当您在其他服务中执行某些操作时，服务代表您请求持有者令牌。此键不适用于 Amazon STS assume-role 操作。

某些 Amazon 服务需要您有权获取 Amazon STS 服务持有者令牌，然后您才能以编程方式访问它们的资源。例如，Amazon CodeArtifact 要求主体使用持有者令牌来执行某些操作。aws `codeartifact get-authorization-token` 命令返回一个持有者令牌。然后，您可以使用持有者令牌来执行 Amazon CodeArtifact 操作。有关持有者令牌的更多信息，请参阅 服务持有者令牌。

sts:IdentityTokenAudience

与字符串运算符结合使用。

使用此键指定 IAM 主体可以使用 GetWebIdentityToken API 向其请求 JSON Web 令牌（JWT）的受众。IAM 策略中存在此条件键时，IAM 主体只能为策略中指定的受众请求令牌。外部服务会验证 JSON Web 令牌中的受众（“aud”）声明，以确保该令牌是为其准备的。

可用性：此键存在于 GetWebIdentityToken API 的请求中，该 API 用于获取通过外部服务进行身份验证的 JSON Web 令牌（JWT）。

在策略中使用此条件键时，请指定与目标收件人的标识符匹配的受众值（例如 https://api.example.com）。

以下示例策略将允许主体请求指定外部服务的令牌：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:GetWebIdentityToken",
            "Resource": "*",
            "Condition": {
                "ForAnyValue:StringEquals": {
                    "sts:IdentityTokenAudience": [
                        "https://api2.example.com",
                        "https://api1.example.com"
                    ]
                }
            }
        }
    ]
}
```

sts:SigningAlgorithm

与字符串运算符结合使用。

使用此键指定 Amazon Amazon STS 用于签署 GetWebIdentityToken API 生成的 JSON Web 令牌（JWT）的加密算法。在策略中使用此条件键时，请指定 ES384（带有 P-384 曲线和 SHA-384 的 ECDSA）或 RS256（带有 SHA-256 的 RSA）。

可用性：此键存在于 GetWebIdentityToken API 的请求中，该 API 用于获取通过外部服务进行身份验证的 JSON Web 令牌（JWT）。

您可以使用此条件键强制要求 IAM 主体使用与您的安全要求或与您集成的外部服务兼容的签名算法来请求令牌。ES384 提供最佳安全性和性能，而 RS256 则提供与不支持 ECDSA 的系统更广泛的兼容性。

以下示例策略要求主体使用 ES384 签名算法：

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "sts:GetWebIdentityToken",
```

```
            "Resource": "*",
            "Condition": {
                "StringEquals": {
                    "sts:SigningAlgorithm": "ES384"
                }
            }
        }
    ]
}
```

sts:ExternalId

与字符串运算符结合使用。

使用此键可要求主体在代入 IAM 角色时提供特定标识符。

可用性 - 如果主体在使用 Amazon CLI 或 Amazon API 代入角色时提供外部 ID，则请求中存在此键。

在其他账户中担任角色时可能需要的唯一标识符。如果角色所属的账户的管理员为您提供了外部 ID，请在 `ExternalId` 参数中提供该值。该值可以是任意字符串，如密码或账号。外部 ID 的主要功能是解决并防止混淆代理人问题。有关外部 ID 和混淆代理人问题的更多信息，请参阅访问第三方拥有的 Amazon Web Services 账户。

`ExternalId` 值的长度必须最少为 2 个字符，最多为 1224 个字符。该值必须是字母数字，没有空格。它还可以包含以下符号：加号 (+)、等号 (=)、逗号 (,)、句点 (.)、@ 符号、冒号 (:)、正斜杠 (/) 和连字符 (-)。

sts:RequestContext/context-key

与字符串运算符结合使用。

使用该键可将在请求中传递的可信令牌颁发者签名上下文断言中嵌入的会话上下文键值对与角色信任策略中指定的上下文键值进行比较。

可用性 – 如果在使用 Amazon STS AssumeRole API 操作代入了角色，则在 `ProvidedContexts` 请求参数中提供上下文断言时，该请求中存在该键。

此上下文键的格式为 `"sts:RequestContext/context-key":"context-value"`，其中 `context-key` 和 `context-value` 是上下文键值对。在请求中传递的已签名上下文断言中嵌入多个上下文键时，每个键值对都有一个上下文键。您必须在角色信任策略中授予 `sts:SetContext` 操作的权限，才能允许主体在生成的会话令牌中设置上下文键。要详细了解可与此键一起使用的支

持的 IAM Identity Center 上下文键，请参阅Amazon IAM Identity Center用户指南中的 IAM Identity Center 的 Amazon STS 条件键。

您可以在角色信任策略中使用该键，以在用户代入角色时根据用户或其属性实施精细的访问控制。代入该角色后，活动将在 Amazon CloudTrail 日志的 `AdditionalEventData` 属性中显示，其中包含由上下文提供程序在代入角色请求中设置的会话上下文键值对。这样，当不同的主体使用角色时，管理员可以更轻松地区分角色会话。键值对由指定的上下文提供程序设置，而不是由 Amazon CloudTrail 或 Amazon STS 设置。这使上下文提供程序可以控制 CloudTrail 日志和会话信息中包含哪些上下文。

sts:RequestContextProviders

与 ARN 运算符结合使用。

使用该键可将请求中的上下文提供程序 ARN 与角色信任策略中指定的上下文提供程序 ARN 进行比较。

可用性 – 如果在使用 Amazon STS AssumeRole API 操作代入了角色，则在 `ProvidedContexts` 请求参数中提供上下文断言时，该请求中存在该键。

以下示例条件检查请求中传递的上下文提供程序 ARN 是否与角色信任策略条件中指定的 ARN 匹配。我们建议您使用 `ForAllValues` 添加一个空检查，以防止缺少上下文键或具有空值的上下文键的计算结果为 true。有关更多信息，请参阅 用于检查条件键是否存在的条件运算符。

JSON

```
{
  "Version":"2012-10-17",
  "Statement": {
    "Action": "sts:SetContext",
    "Effect": "Allow",
    "Resource": "*",
    "Condition": {
      "ForAllValues:ArnEquals": {
        "sts:RequestContextProviders": [
          "arn:aws:iam::aws:contextProvider/IdentityCenter"
        ]
      },
      "Null": {
        "sts:RequestContextProviders": "false"
      }
    }
```

```
  }
}
```

sts:RoleSessionName

与字符串运算符结合使用。

使用此键可将主体在代入角色时指定的会话名称与策略中指定的值进行比较。

可用性 - 当主体使用 Amazon Web Services 管理控制台 管理控制台、assume-role CLI 命令或 Amazon STS `AssumeRole` API 操作代入角色时，则请求中存在此键。

您可以在角色信任策略中使用此键，以要求您的用户在代入角色时提供特定的会话名称。例如，您可以要求 IAM 用户指定自己的用户名作为其会话名称。在 IAM 用户代入角色后，活动将与匹配用户名的会话名称一起显示在 Amazon CloudTrail 日志中。这样，当不同的主体使用角色时，管理员可以更轻松地区分角色会话。

以下角色信任策略要求账户 `111122223333` 中的 IAM 用户在代入角色时提供其 IAM 用户名作为会话名称。使用条件键中的 `aws:username` 条件变量来强制执行此要求。此策略允许 IAM 用户代入策略附加到的角色。此策略禁止任何使用临时凭证的人员代入角色，因为 `username` 变量仅适用于 IAM 用户。

> **Important**
>
> 您可以使用任何可用的单值条件密钥作为变量。您不能使用多值条件键作为变量。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "RoleTrustPolicyRequireUsernameForSessionName",
            "Effect": "Allow",
            "Action": "sts:AssumeRole",
            "Principal": {"AWS": "arn:aws:iam::111122223333:root"},
            "Condition": {
                "StringLike": {"sts:RoleSessionName": "prefix-
${aws:username}"}
```

```
            }
        }
    ]
}
```

当管理员查看操作的 Amazon CloudTrail 日志时，他们可以将会话名称与其账户中的用户名进行比较。在以下示例中，名为 `matjac` 的用户使用名为 `MateoRole` 的角色执行操作。之后，管理员会联系具有名为 `matjac` 用户的 Mateo Jackson。

```
"assumedRoleUser": {
    "assumedRoleId": "AROACQRSTUVWRAOEXAMPLE:matjac",
    "arn": "arn:aws:sts::111122223333:assumed-role/MateoRole/matjac"
}
```

如果您允许使用角色进行跨账户访问，则一个账户中的用户可以代入另一个账户中的角色。CloudTrail 中列出的代入角色的 ARN 包括角色所在的账户。它不包括已代入角色的用户的账户。用户仅在账户中是唯一的。因此，我们建议您使用此方法以仅在 CloudTrail 日志中查看您管理的账户中的用户所代入的角色。您的用户可以在多个账户中使用相同的用户名。

sts:SourceIdentity

与字符串运算符结合使用。

使用此键可将主体在代入角色时指定的源身份与策略中指定的值进行比较。

可用性 — 如果主体提供源身份，同时担任使用任何 Amazon STS 担任角色 CLI 命令或 Amazon STS `AssumeRole` API 操作时，此密钥将会出现在请求中。

您可以在角色信任策略中使用此键，以要求您的用户在代入角色时设置特定的源身份。例如，您可以要求您的工作人员或联合身份为源身份指定值。您可以将身份提供者 (IdP) 配置为使用与用户关联的属性之一（例如用户名或电子邮件）作为源身份。然后，IdP 会将源身份作为其发送的断言或声明中的一个属性传递给 Amazon。源身份属性的值标识担任角色的用户或应用程序。

在用户担任角色后，活动将在使用已设置源身份值的 Amazon CloudTrail 日志中出现。这使管理员能够更轻松地确定什么角色在 Amazon 中执行了操作。您必须授予 `sts:SetSourceIdentity` 操作相应权限以允许身份设置源身份。

与 `sts:RoleSessionName` 不同，在设置源身份后，无法更改该值。它将存在于源身份对角色执行的所有操作的请求上下文中。当您使用会话凭证担任另一个角色时，该值将保留到后续角色会话中。从一个角色代入另一个角色的过程称为角色链。

您可以使用 aws:SourceIdentity 全局条件键根据后续请求中源身份的值进一步控制对 Amazon 资源的访问权限。

以下角色信任策略允许 IAM 用户 AdminUser 在账户中担任角色 111122223333。它还会向 AdminUser 授予权限来设置源身份，只要源身份设置为 DiegoRamirez 即可。

JSON

```
{
    "Version":"2012-10-17",
    "Statement": [
        {
            "Sid": "AllowAdminUserAssumeRole",
            "Effect": "Allow",
            "Principal": {"AWS": " arn:aws:iam::111122223333:user/AdminUser"},
            "Action": [
                "sts:AssumeRole",
                "sts:SetSourceIdentity"
            ],
            "Condition": {
                "StringEquals": {"sts:SourceIdentity": "DiegoRamirez"}
            }
        }
    ]
}
```

要了解有关使用源身份信息的更多信息，请参阅 监控和控制使用所担任角色执行的操作。

sts:TaskPolicyArn

与 ARN 运算符结合使用。

使用该键可将 sts:AssumeRoot 请求中的策略 ARN 与在策略中指定的策略 ARN 进行比较。

可用性：当您使用 sts:AssumeRoot 发出请求时，此键会出现在请求中。

管理员可以在 IAM 策略中使用此条件键，限制管理账户或委托管理员账户中的特定角色或用户使用根凭证执行某些操作。有关更多信息，请参阅 在 Amazon Organizations 成员账户上执行特权任务。

sts:TransitiveTagKeys

与字符串运算符结合使用。

使用此键将请求中的可传递会话标签键，与在策略中指定的键进行比较。

Availability（可用性）- 使用临时安全凭证发出请求时，请求中存在此密钥。这些凭证包括使用任何 assume-role 操作或 `GetFederationToken` 操作创建的凭证。

当您使用临时安全凭证发出请求时，请求上下文包含 `aws:PrincipalTag` 上下文密钥。此键包括会话标签、可传递会话标签和角色标签的列表。可传递会话标签是当您使用会话凭证代入另一个角色时，持续传递到所有后续会话中的标记。从一个角色代入另一个角色的过程称为角色链。

您可以在策略中使用此条件键，以便要求在代入角色或联合身份用户身份时，要求将特定会话标签设置为可传递标签。

Amazon 服务的操作、资源和条件键

每个Amazon服务可以定义操作、资源和条件上下文密钥以在 IAM policy 中使用。有关 Amazon 服务及其操作、资源和条件上下文密钥的列表，请参阅服务授权参考中的操作、资源和条件键。

了解 IAM 详情的资源

IAM 是一款功能丰富的产品，您可找到许多资源来了解有关 IAM 如何帮助保护 Amazon Web Services 账户 和资源的更多信息。

主题

- 身份
- 证书（密码、访问密钥和 MFA 设备）
- 权限与策略
- 联合和委托
- IAM 和其他 Amazon 产品
- 一般安全实践
- 一般 资源

身份

请参阅这些资源，以便创建、管理和使用身份。

- 在 IAM Identity Center 管理身份 – 有关在 IAM Identity Center 中创建用户和组的程序信息。
- IAM 身份 – 有关用户、组和角色的深入讨论。

证书（密码、访问密钥和 MFA 设备）

查看以下指南，为您的 Amazon Web Services 账户 和 IAM 用户管理密码、访问密钥以及 MFA 设备。

- Amazon 中的用户密码 – 介绍为账户中的 IAM 用户管理密码的选项。
- 管理 IAM 用户的访问密钥。 – 介绍访问密钥的工作原理以及如何使用它们以编程方式调用 Amazon。我们建议您首先考虑使用其他更安全的访问密钥的替代方法。有关更多信息，请参阅《Amazon Web Services 一般参考指南》中的 长期访问密钥的注意事项和替代方案。
- IAM 中的 Amazon 多重身份验证 - 介绍如何配置账户和 IAM 用户以要求在允许登录之前提供密码和一次性使用代码（在设备上生成）。（这有时称为双重身份验证）。

有关用于访问 Amazon Web Services 的凭证类型的一般信息，请参阅《Amazon Web Services 一般参考指南》中的 Amazon 安全凭证。

权限与策略

了解 IAM policy 的内部工作原理并查找有关授予权限的最佳方式的提示：

- Amazon Identity and Access Management 中的策略和权限 - 介绍用于定义权限的策略语言。介绍如何将权限附加到用户或组或是 (对于一些 Amazon 产品) 资源本身。
- IAM JSON 策略元素参考 - 提供每个策略语言元素的说明和示例。
- IAM 策略验证 – 查找用于 JSON 策略验证的资源。
- IAM 基于身份的策略示例 - 说明了用于各种 Amazon 产品中的常见任务的策略示例。
- Amazon 策略生成器 - 通过从列表选择产品和操作来创建自定义策略。
- IAM policy simulator - 测试策略是允许还是拒绝对 Amazon 的特定请求。

联合和委托

您可以为在其他位置进行了身份验证（登录）的用户授予对您 Amazon Web Services 账户中的资源的访问权限。这些可以是其他 Amazon Web Services 账户 中的 IAM 用户（称为委派）、使用您企业的登录过程进行了身份验证的用户或是来自互联网身份提供程序 [如 Login with Amazon、Facebook、Google 或任何其他与 OpenID Connect（OIDC）兼容的身份提供程序] 的用户。在这些情况下，用户可获取临时安全凭证以访问 Amazon 资源。

- IAM 教程：使用 IAM 角色委托跨 Amazon 账户的访问权限 – 指导您向其他 Amazon Web Services 账户 中的 IAM 用户授予跨账户访问权限。
- 临时凭证的常见情形 - 介绍在 Amazon 外部进行身份验证之后使用户经过联合身份验证进入 Amazon 的方式。

IAM 和其他 Amazon 产品

大多数 Amazon 产品与 IAM 集成，以便您可以使用 IAM 功能帮助保护对这些产品中的资源的访问。以下资源讨论 IAM 以及一些最受欢迎的 Amazon 产品的安全性。有关使用 IAM 的产品的完整列表（包括有关每个产品的更多信息的链接），请参阅 使用 IAM 的 Amazon 服务。

将 IAM 与 Amazon EC2 结合使用

- 控制对 Amazon EC2 资源的访问 - 介绍如何使用 IAM 功能允许用户管理 Amazon EC2 实例、卷等。

- 使用实例配置文件 - 介绍如何使用 IAM 角色为在 Amazon EC2 实例上运行的应用程序以及需要访问其他 Amazon 产品的应用程序安全地提供凭证。

将 IAM 与 Amazon S3 结合使用

- 管理对 Amazon S3 资源的访问权限 - 讨论用于存储桶和对象（包括 IAM policy）的 Amazon S3 安全模型。
- 编写 IAM policy：授予 Amazon S3 存储桶中用户特定文件夹的访问权限 - 讨论如何让用户在 Amazon S3 中保护自己的文件夹。（有关 Amazon S3 和 IAM 的更多文章，请在博客文章标题下选择 S3 标签。）

将 IAM 与 Amazon RDS 结合使用

- 使用 Amazon Identity and Access Management (IAM) 来管理对 Amazon RDS 资源的访问 — 介绍如何使用 IAM 控制对数据库实例、数据库快照等的访问。
- RDS 资源级别权限读本 - 介绍如何使用 IAM 控制对特定 Amazon RDS 实例的访问。

将 IAM 与 Amazon DynamoDB 结合使用

- 使用 IAM 控制对 DynamoDB 资源的访问 - 介绍如何使用 IAM 允许用户管理 DynamoDB 表和索引。

一般安全实践

查找专家提示和指南，了解保护 Amazon Web Services 账户 和资源的最佳方式：

- 安全、身份和合规性的最佳实践 – 深入介绍如何跨 Amazon Web Services 账户 和产品管理安全性，包括有关安全架构、IAM 的使用、加密和数据安全等方面的建议。
- 身份与访问权限管理 – 该 Amazon Well-Architected Framework 可帮助您了解在云中设计和运行工作负载的关键概念、设计原则和架构最佳实践。
- IAM 的安全防御最佳实操 – 提供相关建议，帮助您了解使用 IAM 保护 Amazon Web Services 账户 和资源的方法。
- Amazon CloudTrail 用户指南 - 使用 Amazon CloudTrail 可跟踪对 Amazon 进行的 API 调用的历史记录并将这些信息存储在日志文件中。这有助于确定访问了您账户中的资源的用户和账户、进行调用的时间、请求的操作等。

一般 资源

浏览以下资源可了解有关 IAM 和 Amazon 的更多信息。

- IAM 的产品信息 - 有关 Amazon Identity and Access Management 产品的一般信息。
- 适用于 Amazon Identity and Access Management 的 Amazon Web Services re:Post – 访问 Amazon Web Services re:Post，与 Amazon 社区讨论涉及 IAM 的技术问题。

- 入门资源中心 – 了解如何设置、Amazon Web Services 账户加入 Amazon 社区和启动您的第一个应用程序。
- Amazon Web Services 支持 中心：用于创建和管理 Amazon Web Services 支持 案例的中心。还提供指向其他有用资源的链接，如论坛、技术常见问题、服务运行状况以及Amazon Trusted Advisor。
- Amazon Web Services 支持 – 提供有关 Amazon Web Services 支持 的信息的主要网页，这是一个一对一的快速响应支持渠道，可以帮助您在云中构建和运行应用程序。
- 联系我们：用于查询有关 Amazon 账单、账户、事件、滥用和其他问题的中央联系点。
- Amazon 网站条款：有关我们的版权和商标、您的账户、许可、网站访问和其他主题的详细信息。

使用 HTTP 查询请求调用 IAM API

内容

- 端点
- 必须使用 HTTPS
- 签署 IAM API 请求

您可以使用查询 API 以编程方式访问 IAM 和 Amazon STS 服务。查询 API 请求是必须包含参数 `Action` 以指示要执行的操作的 HTTPS 请求。IAM 和 Amazon STS 支持所有操作的 GET 和 POST 请求。也就是说，API 不要求您使用某些操作的 GET 请求和其他操作的 POST 请求。然而，GET 请求受 URL 的大小限制；此限制与浏览器相关，通常为 2048 字节。因此，对于要求更高的查询 API 请求，您必须使用 POST 请求。

响应是 XML 文档。有关响应的详细信息，请参阅 IAM 参考或 Amazon Security Token Service API 参考中的单个操作页面。

> **Tip**
>
> 您可以使用 Amazon SDK 之一，而不是直接调用 IAM 或 Amazon STS API 操作。Amazon SDK 中包含适用于各种编程语言和平台（Java、Ruby、.NET、iOS、Android 等）的库和示例代码。开发工具包提供了可通过编程方式访问 IAM 和 Amazon 的便捷方式。例如，这些开发工具包负责处理各种任务，例如以加密方式签署服务请求 (见下文)、管理错误以及自动重试请求。有关 Amazon 软件开发工具包的信息（包括如何下载及安装），请参阅适用于 Amazon Web Services 的工具页面。

有关这些 API 操作和错误的详细信息，请参阅 IAM 参考或 Amazon Security Token Service API 参考。

端点

IAM 和 Amazon STS 各有一个全球终端节点：

- (IAM) https://iam.amazonaws.com
- (Amazon STS) https://sts.amazonaws.com

> **Important**
>
> 除了全局端点，Amazon STS 还支持将请求发送到区域端点。Amazon 建议使用区域端点来减少延迟，建立冗余以及提高会话令牌有效性，而不是使用全局端点。必须先在一个区域中为您的 Amazon Web Services 账户 激活 STS，然后才能在该区域中使用 Amazon STS。有关为 Amazon STS 激活其他区域的更多信息，请参阅管理 Amazon Web Services 区域中的 Amazon STS。

有关所有服务的 Amazon 端点和区域的更多信息，请参阅《Amazon Web Services 一般参考》中的 服务端点和限额。

必须使用 HTTPS

由于查询 API 返回安全凭证等敏感信息，必须对所有 API 请求使用 HTTPS 。

签署 IAM API 请求

必须使用访问密钥 ID 和秘密访问密钥签署请求。我们强烈建议您不要使用 Amazon Web Services 账户根用户凭证处理 IAM 日常工作。您可以使用 IAM 用户凭证，也可以使用 Amazon STS 生成临时安全凭证。

如需对 API 请求进行签名，建议您使用 Amazon 签名版本 4。有关使用签名版本 4 的信息，请转到 Amazon 常规参考 中的签名版本 4 签名流程。

如需使用签名版本 2，可以在 Amazon 常规参考中获取有关签名版本 2 使用方法的信息。

有关更多信息，请参阅下列内容：

- Amazon 安全凭证。提供有关用于访问 Amazon 的凭证类型的一般信息。
- IAM 的安全防御最佳实操.提出一系列使用 IAM 服务帮助保护 Amazon 资源的建议。
- IAM 临时安全凭证.说明如何创建和使用临时安全凭证。

IAM 文件历史记录

下表介绍了 IAM 的主要文档更新。

变更	说明	日期
job_workflow_ref 已被接受为 GitHub Actions OIDC 提供程序的替代作用域声明	对于 GitHub Actions OIDC 提供程序，IAM 现在在接受角色信任策略中的身份提供程序控制时，接受 sub 声明或 job_workflow_ref 声明。	2026 年 5 月 12 日
特定服务凭证 API 的 IAM 条件上下文键	IAM 现在支持两个新的条件上下文键，用于控制特定服务凭证 API 的访问权限：iam:ServiceSpecificCredentialAgeDays 和 iam:ServiceSpecificCredentialServiceName 。通过这些键，您可以根据到期设置和允许的 Amazon 服务限制特定服务凭证的创建和管理。 • iam:ServiceSpecificCredentialAgeDays • iam:ServiceSpecificCredentialServiceName	2025 年 9 月 4 日
用于网络边界控制的 VPC 端点条件键	IAM 现在支持三个新的 Amazon 全局条件上下文键，用于实现可扩展的网络边界控制：aws:VpceAccount 、aws:VpceOrgID 和 aws:VpceOrgPaths 。这些键有助于确保请求经由特定	2025 年 8 月 28 日

	账户、组织或组织单元拥有的 VPC 端点，可根据 VPC 端点的使用情况自动扩展，且在创建新端点时无需更新策略。有关更多信息，请参阅使用数据边界建立权限防护机制。	
IAM SAML 联合身份验证 Amazon CloudFormation 教程	IAM 添加了有关使用 Amazon CloudFormation 创建 SAML 身份提供者 (IdP) 和联合角色的新教程。 • 使用 Amazon CloudFormation 创建 SAML IdP • 使用 Amazon CloudFormation 创建 SAML 联合角色 • 使用 Amazon CloudFormation 创建 SAML IdP 和联合角色	2025 年 8 月 25 日
适用于共享 OIDC 提供者的其他共享 OIDC 提供者控制	IAM 现在将 Amazon Cognito、Azure Sentinel、Pulumi Cloud 和 Vercel 全局端点纳入了需要显式评估 JSON Web 令牌（JWT）中特定声明的共享 OIDC 身份提供者列表。	2025 年 8 月 1 日
IAM Access Analyzer 添加了内部访问分析器	IAM Access Analyzer 帮助识别您的组织或账户内的哪些主体有权访问所选关键业务资源。内部访问分析器支持实施最低权限原则，确保只有您的 Amazon 组织或账户内的预期主体才能访问指定的资源。	2025 年 6 月 16 日

适用于共享 OIDC 提供者的身份提供者控制	IAM 现在要求对认可的共享 OIDC 身份提供者的 JSON Web 令牌（JWT）中的特定声明进行明确评估。此安全控制可以确保，只有预期组织的授权身份才能代入角色和访问 Amazon 资源。	2025 年 6 月 6 日
IAMUserChangePassword ：添加了权限	IAM 向 IAMUserChangePassword 添加了权限，以允许在路径内指定的用户。	2025 年 5 月 28 日
AccessAnalyzerServiceRolePolicy ：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了 `iam:GetAccountAuthorizationDetails` 。	2025 年 5 月 12 日
AccessAnalyzerServiceRolePolicy ：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了 Amazon S3 目录存储桶接入点。	2025 年 3 月 31 日
IAM 双堆栈端点支持	IAM 现在提供改进的双堆栈端点支持，使客户端能够使用 IPv4 或 IPv6 地址与 IAM 通信。	2025 年 3 月 20 日
更新策略评估逻辑	增强了策略评估逻辑的文档，包括改进的流程图以及对 Amazon 如何评估策略以确定是允许还是拒绝请求的更清晰解释。	2025 年 1 月 30 日
IAMDeleteRootUserCredentials – 移除的权限	IAM 从托管策略中移除了 `iam:DeleteVirtualMFADevice` 权限。	2025 年 1 月 7 日

AccessAnalyzerServiceRolePolicy ：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限新增了对检索 Amazon ECR 账户设置和注册表策略信息的权限的支持。	2024 年 12 月 10 日
IAM Access Analyzer 增加了访问配置	IAM Access Analyzer 增加了对配置分析器的支持，以更改 Amazon Web Services 账户、IAM 用户和角色生成调查发现的范围。	2024 年 11 月 14 日
集中管理成员账户的根访问权限	现在，您可以通过集中的根访问在 Amazon Organizations 中管理跨成员账户的特权根用户凭证。集中保护使用 Amazon Organizations 管理的 Amazon Web Services 账户的根用户凭证，并移除和防止根用户凭证恢复和大规模访问。 • 根用户最佳实践	2024 年 11 月 14 日
Amazon 托管式策略更新 - 新策略	IAM 添加了两个新策略来限制特权根用户会话的权限范围，您可以在集中组织中成员账户的根用户访问后启动这些会话。 • IAMAuditRootUserCredentials • IAMCreateRootUserPassword • IAMDeleteRootUserCredentials • S3UnlockBucketPolicy • SQSUnlockQueuePolicy	2024 年 11 月 14 日

对 Amazon Organizations 资源控制策略（RCP）的支持	使用 Amazon Organizations 资源控制策略（RCP）为组织或组织单位（OU）中账户内的资源定义最大权限。RCP 限制了基于身份和基于资源的策略可以向组织内账户中的资源授予的权限。	2024 年 11 月 13 日
AccessAnalyzerServiceRolePolicy：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对检索 IAM 用户和角色标签信息的权限的支持。	2024 年 10 月 29 日
SAML 加密支持增强功能	增强了 IAM SAML 提供商中 SAML 加密支持的文档，包括改进的故障排除指南和服务兼容性澄清。	2024 年 6 月 5 日
AccessAnalyzerServiceRolePolicy：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对检索 IAM 用户和角色策略信息的权限的支持。	2024 年 5 月 30 日
对 SAML 身份提供商的加密支持	IAM SAML 提供商现在支持在来自外部 IdP 的 SAML 响应中进行加密断言。要了解加密如何与 IAM SAML 联合身份验证配合使用，请参阅使用基于 SAML 的联合身份验证进行 API 访问。	2024 年 2 月 4 日

AccessAnalyzerServiceRolePolicy：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了对检索阻止公开访问 Amazon EC2 快照当前状态的权限的支持。	2024 年 1 月 23 日
AccessAnalyzerServiceRolePolicy：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了 DynamoDB 流和表。	2024 年 1 月 11 日
AccessAnalyzerServiceRolePolicy：添加了权限	IAM Access Analyzer 向 AccessAnalyzerServiceRolePolicy 的服务级别权限添加了 Amazon S3 目录存储桶。	2023 年 12 月 1 日
IAMAccessAnalyzerReadOnlyAccess：添加了权限	IAM Access Analyzer 向 IAMAccessAnalyzerReadOnlyAccess 添加了权限，让您可以检查策略更新是否授予额外的访问权限。 IAM Access Analyzer 需要此权限才能对您的策略执行策略检查。	2023 年 11 月 26 日
IAM Access Analyzer 添加了未使用的访问分析器	IAM Access Analyzer 简化了对未使用访问的检查，以指导您获得最低权限。IAM Access Analyzer 会持续分析您的账户，以识别未使用的访问，并创建一个包含调查发现的集中式控制面板。	2023 年 11 月 26 日
IAM Access Analyzer 添加了自定义策略检查	IAM Access Analyzer 现在提供了自定义策略检查，可在部署之前验证 IAM policy 是否符合您的安全标准。	2023 年 11 月 26 日

AccessAnalyzerServiceRolePolicy ：添加了权限	IAM Access Analyzer 将 IAM 操作添加到 AccessAnalyzerServiceRolePolicy 的服务级别权限，以支持以下操作： • 列出策略的实体 • 生成服务上次访问的详细信息 • 列出访问密钥信息	2023 年 11 月 26 日
上次访问操作的信息和支持针对 60 多项其他服务和操作生成策略	IAM 现在支持上次访问操作的信息，并针对超过 60 项其他服务生成包含操作级信息的策略，并提供上次访问操作信息的操作列表。	2023 年 11 月 1 日
针对 140 多项服务提供上次访问操作的信息支持	IAM 现在提供针对 140 多项服务的上次访问操作的信息，并提供上次访问操作的信息的操作列表。	2023 年 9 月 14 日
支持多个多重身份验证（MFA）设备用于根用户和 IAM 用户	现在，您最多可以为每个用户添加八个 MFA 设备，包括 FIDO 安全密钥、带有虚拟身份验证应用程序的基于时间的一次性密码（TOTP）或硬件 TOTP 令牌。	2022 年 11 月 16 日

IAM Access Analyzer 支持新的资源类型	IAM Access Analyzer 附加支持以下资源类型： • Amazon EBS 卷快照 • Amazon ECR 存储库 • Amazon EFS 文件系统 • Amazon RDS 数据库快照 • Amazon RDS 数据库集群快照 • Amazon SNS 主题	2022 年 10 月 25 日
U2F 弃用和 WebAuthn/FIDO 更新	删除了提及 U2F 作为 MFA 选项的内容，并添加了有关 WebAuthn、FIDO2 和 FIDO 安全密钥的信息。	2022 年 5 月 31 日
IAM 中恢复能力的更新	新增了有关在事件中断 Amazon Web Services 区域 之间的通信时保持对 IAM 凭证的访问权限的信息。	2022 年 5 月 16 日
新的资源全局条件键	现在，您可以根据 Amazon Organizations 中的账户、组织单位（OU）或组织来控制对包含您的资源的资源的访问权限。您可以使用 IAM policy 中的 `aws:ResourceAccount` 、`aws:ResourceOrgID` 和 `aws:ResourceOrgPaths` 全局条件键。	2022 年 4 月 27 日

使用 Amazon 开发工具包的 IAM 代码示例	新增的代码示例显示如何将 IAM 与 Amazon 软件开发工具包（SDK）一起使用。这些示例被划分成代码摘录，展示如何调用单个服务函数，以及展示如何通过在同一服务中调用多个函数来完成特定任务。	2022 年 4 月 7 日
更新策略评估逻辑流程图	更新确定账户中是允许还是拒绝请求部分中的策略评估逻辑流程图和相关文本。	2021 年 11 月 17 日
安全最佳实践的更新	添加了有关创建管理用户而不是使用根用户凭证的信息，移除了使用 IAM 组向 IAM 用户分配权限的最佳实践，并澄清了何时使用托管策略而不是内联策略。	2021 年 10 月 5 日
基于资源的策略的策略评估逻辑主题的更新	添加了有关同一账户中基于资源的策略和不同主体类型的影响的信息。	2021 年 10 月 5 日
对单值和多值条件密钥的更新	现在将更详细地解释单值和多值条件键之间的差异。值类型已添加到每个 Amazon 全局条件上下文密钥。	2021 年 9 月 30 日
IAM Access Analyzer 支持 Amazon S3 多区域访问点	IAM Access Analyzer 可识别允许公共和跨账户访问的 Amazon S3 存储桶，包括使用 Amazon S3 多区域访问点的存储桶。	2021 年 9 月 2 日
Amazon 管理策略更新 — 对现有策略的更新	IAM Access Analyzer 更新了现有 Amazon 管理策略。	2021 年 9 月 2 日

支持更多用于生成操作级策略的服务	IAM Access Analyzer 可以为其他 Amazon 服务生成包含操作级访问活动信息的 IAM policy。	2021 年 8 月 24 日
为跨账户跟踪生成 IAM policy	现在，您可以使用 IAM Access Analyzer 根据您在不同账户中使用 Amazon CloudTrail 跟踪的访问活动生成精细策略，例如集中的 Amazon Organizations 跟踪。	2021 年 8 月 18 日
其他 IAM Access Analyzer 策略检查	IAM Access Analyzer 通过添加新的策略检查来验证 IAM policy 中包含的条件，从而扩展策略验证。这些检查会分析策略语句中的条件数据块，并报告安全警告、错误和建议以及可操作建议。 IAM Access Analyzer 添加了以下策略检查： • 错误 — 无效的服务主体格式 • 错误 — 条件中缺少标签密钥 • 安全警告 — 使用不支持的标签条件键拒绝服务的 NotAction • 安全警告 — 使用不支持的标签条件键拒绝服务 • 安全警告 — 缺少配对条件键 • 建议 — 使用不支持的标签条件键允许服务的 NotAction • 建议 — 使用不支持的标签条件键允许服务	2021 年 6 月 29 日

上次访问的操作支持更多服务	现在，您可以在 IAM 控制台中查看上次访问的操作信息，了解 IAM 主体上次对以下服务使用操作的内容：Amazon EC2、IAM、Lambda 和 Amazon S3 管理操作。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。您可以使用此信息确定不必要的权限，从而优化 IAM policy 以更好地遵循最低权限原则。	2021 年 4 月 19 日
监控和控制使用担任角色执行的操作	管理员可以配置 IAM 角色，以要求身份传递已在 Amazon CloudTrail 中登录的源身份。查看源身份信息可帮助管理员确定谁通过担任角色会话执行了操作或执行了哪些操作。	2021 年 4 月 13 日
基于访问活动生成 IAM policy	现在，您可以使用 IAM Access Analyzer 根据在 Amazon CloudTrail 中找到的访问活动生成精密策略了。	2021 年 4 月 7 日
IAM Access Analyzer 策略检查	IAM Access Analyzer 现在在策略创作过程中提供了 100 多项策略检查，并附带了可操作的建议。	2021 年 3 月 16 日
扩展的策略验证选项	使用 IAM Access Analyzer 中的策略检查扩展 IAM 控制台、Amazon API 以及 Amazon CLI 中可用的策略验证来帮助您编写安全且功能性强的 JSON 策略。	2021 年 3 月 15 日

标记 IAM 资源	您现在可以使用标签键值对标记额外的 IAM 资源。	2021 年 2 月 11 日
IAM 用户的默认密码策略	如果您没有为 Amazon Web Services 账户 设置自定义密码策略，则 IAM 用户密码现在必须符合默认 Amazon 密码策略。	2020 年 11 月 18 日
Amazon 服务的操作、资源和条件键页面已移动	每个 Amazon 服务可以定义操作、资源和条件上下文密钥以在 IAM policy 中使用。有关 Amazon 服务及其操作、资源和条件上下文密钥的列表，请参阅服务授权参考。	2020 年 11 月 16 日
IAM 用户更长的角色会话持续时间	IAM 用户在 Amazon Web Services 管理控制台 中切换角色时现在可以拥有更长的角色会话持续时间，从而减少由于会话过期而造成的中断。用户被授予为角色设置的最大会话持续时间或 IAM 用户会话中的剩余时间（以较少者为准）。	2020 年 7 月 24 日
使用 Service Quotas 请求快速增加 IAM 实体	您可以使用 Service Quotas 控制台请求增加可调 IAM 配额的配额。现在，某些增加将在 Service Quotas 中几分钟内自动得到批准并可用于您的账户中。更大的增加请求将提交给 Amazon Web Services 支持。	2020 年 6 月 25 日

Amazon S3 中的上次访问信息现在包括 IAM 管理操作	除了服务上次访问的信息之外，您现在可以在 IAM 控制台中查看有关 IAM 主体上次使用 Amazon S3 操作的时间的信息。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。该报告包括有关主体上次尝试访问的允许的服务和操作以及访问时间的信息。您可以使用此信息确定不必要的权限，从而优化 IAM policy 以更好地遵循最低权限原则。	2020 年 6 月 3 日
添加了安全性章节	安全性章节可帮助您了解如何配置 IAM 和 Amazon STS 以满足您的安全性和合规性目标。您还会了解如何使用其他 Amazon 服务以帮助您监控和保护 IAM 资源。	2020 年 4 月 29 日
sts:RoleSessionName	您现在可以编写一个策略，该策略基于主体在代入角色时指定的会话名称授予权限。	2020 年 4 月 21 日
Amazon 登录页面更新	在主 Amazon 登录页面上登录时，您现在可以选择以 Amazon Web Services 账户根用户 或 IAM 用户身份登录。当您执行此操作时，页面上的标签会指示您是否应提供根用户电子邮件地址或 IAM 用户信息。本文档包含更新的屏幕截图，以帮助您了解 Amazon 登录页面。	2020 年 3 月 4 日

aws:ViaAWSService 和 aws:CalledVia 条件键	您现在可以编写策略来限制服务是否可以代表 IAM 主体（用户或角色）发出请求。当主体向 Amazon 服务发出请求时，该服务可能会使用主体的凭证向其他服务发出后续请求。如果任何服务使用主体的凭证发出请求，请使用 `aws:ViaAWSService` 条件键进行匹配。如果特定服务使用主体的凭证发出请求，请使用 `aws:CalledVia` 条件键进行匹配。	2020 年 2 月 20 日
策略模拟器添加对权限边界的支持	您现在可以使用 IAM policy simulator 测试权限边界对 IAM 实体的影响。	2020 年 1 月 23 日
跨账户策略评估	现在，您可以了解 Amazon 如何评估跨账户访问策略。当信任账户中的资源包含的基于资源的策略允许另一个账户中的主体访问该资源时，就会发生此情况。这两个账户都必须允许该请求。	2020 年 1 月 2 日
会话标签	现在，您可以在 Amazon STS 中代入角色或联合身份用户身份时包含标签。执行 `AssumeRole` 或 `GetFederationToken` 操作时，您可以将会话标签作为属性传递。执行 `AssumeRoleWithSAML` 或 `AssumeRoleWithWebIdentity` 操作时，您可以将属性从公司身份传递到 Amazon。	2019 年 11 月 22 日

控制 Amazon Organizations 中 Amazon Web Services 账户 组的访问权限	现在，您可以在 IAM policy 中从 Amazon Organizations 引用企业部门 (OU)。如果您使用 Amazon Organizations 将账户组织到 OU 中，则可以要求委托人属于特定 OU，然后再向其授予对您资源的访问权限。主体包括 Amazon Web Services 账户根用户、IAM 用户和 IAM 角色。为此，请在策略的 `aws:PrincipalOrgPaths` 条件键中指定 OU 路径。	2019 年 11 月 20 日
上次使用的角色	您现在可以查看上次使用角色的日期、时间和区域。此信息还可帮助您确定账户中未使用的角色。您可以使用 Amazon Web Services 管理控制台、Amazon CLI 和 Amazon API 查看有关上次使用角色时间的信息。	2019 年 11 月 19 日
对全局条件上下文密钥页面的更新	现在，您可以了解什么时候在请求的上下文中包含各个全局条件键。您还可以使用页面目录 (TOC) 轻松地导航到各个键。页面上的信息可帮助您编写更准确的策略。例如，如果您的员工通过 IAM 角色使用联合身份，则您应使用 `aws:userId` 密钥而不是 `aws:userName` 密钥。`aws:userName` 密钥仅适用于 IAM 用户，不适用于角色。	2019 年 10 月 6 日

Amazon 中的 ABAC	了解如何通过标签在 Amazon 中使用基于属性的访问控制 (ABAC)，以及它与传统 Amazon 授权模型的比较。使用 ABAC 教程，了解如何创建策略，允许具有主体标签的 IAM 角色访问具有匹配标签的资源，并测试该策略。此策略仅允许用户查看或编辑其作业需要的 Amazon 资源。	2019 年 10 月 3 日
Amazon STS GetAccessKeyInfo 操作	您可以在代码中查看 Amazon 访问密钥，以确定密钥是否来自于您拥有的账户。您可以使用 `aws sts get-access-key-info` Amazon CLI 命令或 `GetAccessKeyInfo` Amazon API 操作传递访问密钥 ID。	2019 年 7 月 24 日
在 IAM 中查看 Amazon Organizations 服务的上次访问信息	您现在可以在 IAM 控制台的 Amazon Organizations 部分中查看 Amazon Organizations 实体或策略的上次访问的服务信息。您也可以使用 Amazon CLI 或 Amazon API 检索数据报告。该数据包括有关 Amazon Organizations 账户中的委托人上次尝试访问的允许服务以及访问时间的信息。您可以使用此信息确定不必要的权限，从而优化 Amazon Organizations 策略以更好地遵循最小特权原则。	2019 年 6 月 20 日

将托管策略用作会话策略	您现在可以在担任角色时最多传递 10 个托管策略 ARN。这允许您限制角色的临时凭证的权限。	2019 年 5 月 7 日
全局端点的会话令牌的 Amazon STS 区域兼容性	您现在可以选择是使用版本 1 还是版本 2 全球端点令牌。版本 1 令牌仅在默认启用的 Amazon 区域中有效。这些令牌不适用于手动启用的区域，例如，亚太地区（香港）。版本 2 令牌在所有区域中都有效。不过，版本 2 令牌较长，可能会影响临时存储令牌的系统。	2019 年 4 月 26 日
允许启用和禁用 Amazon 区域	您现在可以创建策略，以允许管理员启用和禁用亚太地区（香港）区域 (ap-east-1)。	2019 年 4 月 24 日
IAM 用户的“我的安全凭证”页面	IAM 用户现在可在 My Security Credentials（我的安全凭证）页面上管理自己的所有凭证。此 Amazon Web Services 管理控制台 页面显示账户信息，例如账户 ID 和规范用户 ID。用户还可以查看和编辑自己的密码、访问密钥、X.509 证书、SSH 密钥和 Git 凭证。	2019 年 1 月 24 日
访问分析器 API	您现在可使用 Amazon CLI 和 Amazon API 查看上次访问的服务相关信息。	2018 年 12 月 7 日

标记 IAM 用户和角色	您现在可使用 IAM 标签利用标签键值对向身份（IAM 用户或角色）添加自定义属性。您还可以使用标签控制身份对资源的访问权限或控制可附加到身份的标签。	2018 年 11 月 14 日
U2F 安全密钥	现在，您可以使用 U2F 安全密钥作为登录 Amazon Web Services 管理控制台 时的 Multi-Factor Authentication (MFA) 选项。	2018 年 9 月 25 日
对 Amazon VPC 端点的支持	美国西部（俄勒冈州）区域中，您现在可以在 VPC 与 Amazon STS 之间建立私有连接。	2018 年 7 月 31 日
权限边界	利用新功能，可以更轻松地向可信员工授予管理 IAM 权限的能力，而无需授予完整的 IAM 管理访问权限。	2018 年 7 月 12 日
aws:PrincipalOrgID	通过指定 IAM 主体的 Amazon 企业，新的条件键提供了控制对 Amazon 资源的访问的更简单方法。	2018 年 5 月 17 日
aws:RequestedRegion	新的条件键提供了使用 IAM policy 来控制对 Amazon 区域的访问的更简单方法。	2018 年 4 月 25 日
增加了 IAM 角色的会话持续时间	IAM 角色 角色现在可以具有 12 小时的会话持续时间。	2018 年 3 月 28 日
更新了角色创建工作流程	新的工作流程改进了创建信任关系并将权限附加到角色的过程。	2017 年 9 月 8 日

Amazon Web Services 账户 登录流程	更新的 Amazon 登录体验允许根用户和 IAM 用户使用 Amazon Web Services 管理控制台 主页上的 Sign In to the Console（登录控制台）链接。	2017 年 8 月 25 日
示例 IAM policy	文档更新包含 30 多个示例策略。	2017 年 8 月 2 日
IAM 最佳实践	利用添加到 IAM 控制台的 Users（用户）部分中的信息，可以更轻松地遵循 IAM 最佳实践。	2017 年 7 月 5 日
Auto Scaling 资源	资源级权限可以控制对 Auto Scaling 资源的访问以及 Auto Scaling 资源的权限。	2017 年 5 月 16 日
Amazon RDS for MySQL 和 Amazon Aurora 数据库	数据库管理员可以将数据库用户与 IAM 用户和角色相关联，从而从一个位置管理对所有 Amazon 资源的用户访问。	2017 年 4 月 24 日
服务关联角色	服务相关角色提供了更轻松、更安全的方法来将权限委派给 Amazon 服务。	2017 年 4 月 19 日
策略摘要	新的策略摘要使您可以更轻松地了解 IAM policy 中的权限。	2017 年 3 月 23 日