亚马逊云科技

开发人员指南

Amazon Cognito

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon Cognito: 开发人员指南

Copyright © 2025 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

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

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

Table of Contents

什么是 Amazon Cognito？ 1
- 用户池 2
- 身份池 3
- Amazon Cognito 的功能 4
  - 用户池 4
  - 身份池 6
- Amazon Cognito 用户池和身份池的比较 7
- Amazon Cognito 入门 10
- 区域可用性 11
- Amazon Cognito 的定价 11
- 术语和概念 11
  - 常规 12
  - 用户池 14
  - 身份池 18
- 入门 Amazon 19
  - 注册获取 Amazon Web Services 账户 19
  - 保护 IAM 用户 19

用户池入门 21
- 您的第一个应用程序和用户池 21
- 其他应用程序选项 23
  - React SPA 示例 24
  - Flutter 移动应用程序示例 27
- 后续步骤 30
  - 添加社交身份提供商 31
  - 添加 SAML IdP 38

身份池入门 41
- 在 Amazon Cognito 中创建一个身份池 41
- 设置 SDK 43
- 集成身份提供商 43
- 获取凭证 43

其他入门选项 44

与应用程序集成 46
- 使用进行身份验证 Amazon Amplify 47
  - 使用 Amplify 创建用户界面（UI） 48

使用进行身份验证 Amazon SDKs 48
身份验证的工作原理 49
托管登录身份验证 49
SDK 身份验证 53
第三方身份提供者身份验证 56
身份池身份验证 59
与 Amazon SDKs 62
使用 Amazon Verified Permissions 进行授权 62
使用 Verified Permissions 进行 API 授权 64
Amazon Cognito 用户的示例策略 67
代码示例 69
Amazon Cognito Identity 70
基本功能 71
场景 93
Amazon Cognito 身份提供者 95
基本功能 105
场景 258
Amazon Cognito Sync 423
基本功能 424
多租赁最佳实践 427
按租户划分的用户池 428
按租户划分的应用程序客户端 430
按租户划分的用户池组 432
按租户划分的自定义属性 434
按租户划分的自定义范围 436
示例资源 438
多租户安全建议 440
Amazon Cognito 常见场景 441
使用用户池进行身份验证 441
访问服务器端资源 442
使用 API Gateway 和 Lambda 来访问资源 442
使用用户池和身份池访问 Amazon 服务 443
借助第三方进行身份验证并使用身份池访问 Amazon 服务 444
使用 Amazon Cognito 访问 Amazon AppSync 资源 445
Amazon Cognito 用户群体 447
特征 448

注册 448
登录 449
托管登录 450
安全性 450
自定义客户体验 450
监控和分析 451
Amazon Cognito 身份池集成 451
用户池功能计划 452
选择功能计划 453
按计划划分的功能 454
基本计划功能 456
加上套餐功能 460
关闭不符合条件的功能 462
安全最佳实践 463
在网络层面保护您的用户池 463
防范短信滥用 464
了解公共身份验证 464
使用客户机密保护机密客户 467
保护其他机密 467
用户池管理最低权限 468
保护和验证令牌 470
确定您要信任的身份提供商 470
了解范围对用户个人资料访问权限的影响 471
清理用户属性的输入 471
身份验证 471
实现身份验证流程 472
需知信息 474
身份验证流程示例 477
托管登录身份验证 480
SDK 身份验证 482
身份验证流程 486
SDK 授权模型 508
第三方 IdP 登录 521
联合登录在 Amazon Cognito 用户群体中的工作方式 522
应用程序作为 Amazon Cognito 的服务提供者的责任 523
关于 Amazon Cognito 用户群体第三方登录需要了解的事项 523

身份提供者 524
社交身份提供者 530
SAML 提供商 537
OIDC 提供者 564
映射 IdP 属性 574
链接联合用户 579
托管登录 582
托管登录本地化 584
使用设置托管登录 Amazon Amplify 585
使用 Amazon Cognito 控制台设置托管登录 585
查看您的登录页面 586
自定义您的身份验证页面 587
关于托管登录和托管用户界面的注意事项 587
配置域 590
品牌和定制 601
使用 Lambda 触发器 619
需知信息 621
设置触发器 623
用户池 Lambda 触发器事件 624
用户池 Lambda 触发器通用参数 625
客户机元数据 626
按操作触发源 628
按功能触发源 634
注册前 639
确认后 645
身份验证前 648
身份验证后 652
自定义挑战 656
令牌生成前 671
迁移用户 693
自定义消息 698
自定义发件人 705
管理用户 723
允许用户注册 724
注册并确认用户账户 726
以管理员身份创建用户 749

向用户池添加组 755
管理和搜索用户 758
密码 765
将用户导入一个用户池 769
Attributes 786
用户池令牌 800
ID 令牌 801
访问令牌 805
刷新令牌 809
撤消令牌 814
验证 JSON Web 令牌 816
管理用户池令牌到期和缓存 821
在登录后访问资源 824
使用 Verified Permissions 访问资源 442
访问 API Gateway 资源 827
使用身份池访问 Amazon 资源 828
其他功能 833
更新用户池和应用程序客户端 833
应用程序客户端 837
使用设备 845
带资源服务器的访问控制 851
使用 Amazon Pinpoint 分析 859
电子邮件设置 864
短信设置 877
使用安全功能 885
添加 MFA 887
威胁防护 904
Amazon WAF Web ACLs 929
区分大小写 934
删除保护 935
管理用户泄露 937
用户池端点参考 943
托管登录端点 944
联合身份验证端点 950
OAuth 2.0 补助金 979
使用 PKCE 980

托管登录和联盟错误响应 982
Amazon Cognito 身份池 984
配置身份池 986
创建 身份池 986
用户 IAM 角色 988
经过身份验证和未经身份验证的身份 988
激活或停用访客访问权限 988
更改与身份类型关联的角色 989
编辑身份提供者 990
删除身份池 991
从身份池删除身份 991
将 Amazon Cognito Sync 与身份池一起使用 992
身份池身份验证流程 994
IAM 角色 1003
设置信任策略 1004
访问策略 1007
角色信任和权限 1017
安全最佳实践 1018
IAM 配置最佳实践 1019
身份池配置最佳实践 1021
将属性用于访问控制 1022
使用属性对 Amazon Cognito 身份池进行访问控制 1023
示例：将属性用于访问控制策略 1024
关闭访问控制属性 1026
默认提供商映射 1026
使用基于角色的访问控制 1028
为角色映射创建角色 1028
授予传递角色权限 1029
使用令牌向用户分配角色 1030
使用基于规则的映射向用户分配角色 1031
基于规则的映射中使用的令牌声明 1032
基于角色的访问控制的最佳实践 1034
获取凭证 1034
使用 凭证 1041
第三方身份提供者 1043
Facebook 1044

Login with Amazon 1051
Google 1055
通过 Apple 登录 1063
Open ID Connect 提供商 1069
SAML 身份提供商 1071
经开发人员验证的身份 1073
了解身份验证流程 1074
定义开发人员提供商名称并将其与身份池关联 1074
实施身份提供商 1075
更新登录映射（仅限 Android 和 iOS） 1083
获取令牌（服务器端） 1084
连接到现有社交身份 1085
支持在提供商之间转换 1085
切换身份 1089
Android 1089
iOS – objective-C 1090
iOS – swift 1090
JavaScript 1091
Unity 1091
Xamarin 1092
Amazon Cognito Sync 1093
Amazon Cognito Sync 入门 1093
设置 Amazon Cognito 中的身份池 1094
存储和同步数据 1094
同步不同客户端的数据 1094
初始化 Amazon Cognito Sync 客户端 1095
了解数据集 1097
在数据集中读取并写入数据 1098
使用同步存储同步本地数据 1100
处理事件回调 1103
Android 1104
iOS - Objective-C 1106
iOS - Swift 1109
JavaScript 1112
Unity 1115
Xamarin 1118

实施推送同步 1120
创建 Amazon Simple Notification Service（Amazon SNS）应用程序 1121
在 Amazon Cognito 控制台中启用推送同步 1121
在您的应用程序中使用推送同步：Android 1122
在您的应用程序中使用推送同步：iOS – Objective-C 1124
在您的应用程序中使用推送同步：iOS – Swift 1126
实施 Amazon Cognito Sync 流 1129
使用 Amazon Cognito Events 自定义工作流 1131
安全性 1136
数据保护 1136
数据加密 1137
身份和访问管理 1138
受众 1138
使用身份进行身份验证 1139
使用策略管理访问 1141
Amazon Cognito 如何与 IAM 配合使用 1143
基于身份的策略示例 1151
故障排除 1155
使用服务相关角色 1157
日志记录和监控 1161
监控成本 1162
导出用户池日志 1164
监控配额和使用情况 1174
CloudTrail 日志 1188
合规性验证 1215
恢复能力 1215
区域数据注意事项 1216
基础结构安全性 1217
配置和漏洞分析 1217
Amazon 托管策略 1217
策略更新 1219
为资源添加标签 1221
支持的资源 1221
标签限制 1221
使用控制台管理标签 1222
Amazon CLI 例子 1222

分配标签 1222
查看标签 1224
删除标签 1224
在创建资源时应用标签 1225
API 操作 1225
适用于用户池标签的 API 操作 1225
适用于身份池标签的 API 操作 1226
限额 1227
了解 API 请求速率配额 1227
配额分类 1227
使用特殊请求速率处理 Amazon Cognito 用户池 API 操作 1228
每月活跃用户 1228
管理 API 请求速率配额 1230
确定配额要求 1230
优化请求速率 1231
跟踪配额使用量 1231
跟踪每月活跃用户 (MAUs) 1232
请求提高限额 1233
用户池请求速率配额 1233
身份池请求速率配额 1244
资源数量和大小的配额 1245
文档历史记录 1252
..... mcclxvii

什么是 Amazon Cognito？

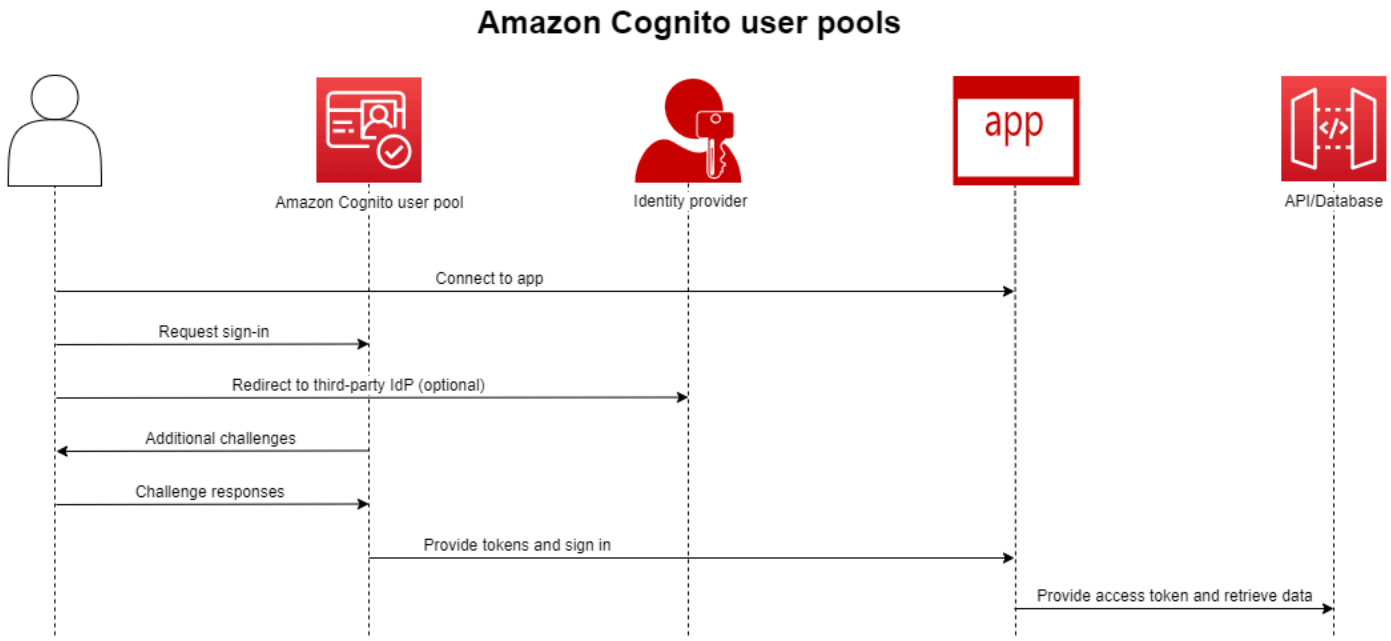
Amazon Cognito 是 Web 和移动应用程序的身份平台。它是一个用户目录、一个身份验证服务器以及一个用于 OAuth 2.0 访问令牌和 Amazon 凭据的授权服务。使用 Amazon Cognito，您可以对内置用户目录、企业目录以及 Google 和 Facebook 等使用者身份提供者中的用户进行身份验证和授权。

主题

- 用户池
- 身份池
- Amazon Cognito 的功能
- Amazon Cognito 用户池和身份池的比较
- Amazon Cognito 入门
- 区域可用性
- Amazon Cognito 的定价
- 常见 Amazon Cognito 术语和概念
- 入门 Amazon

随后的两个组件构成了 Amazon Cognito。它们根据用户的访问需求独立或协同运行。

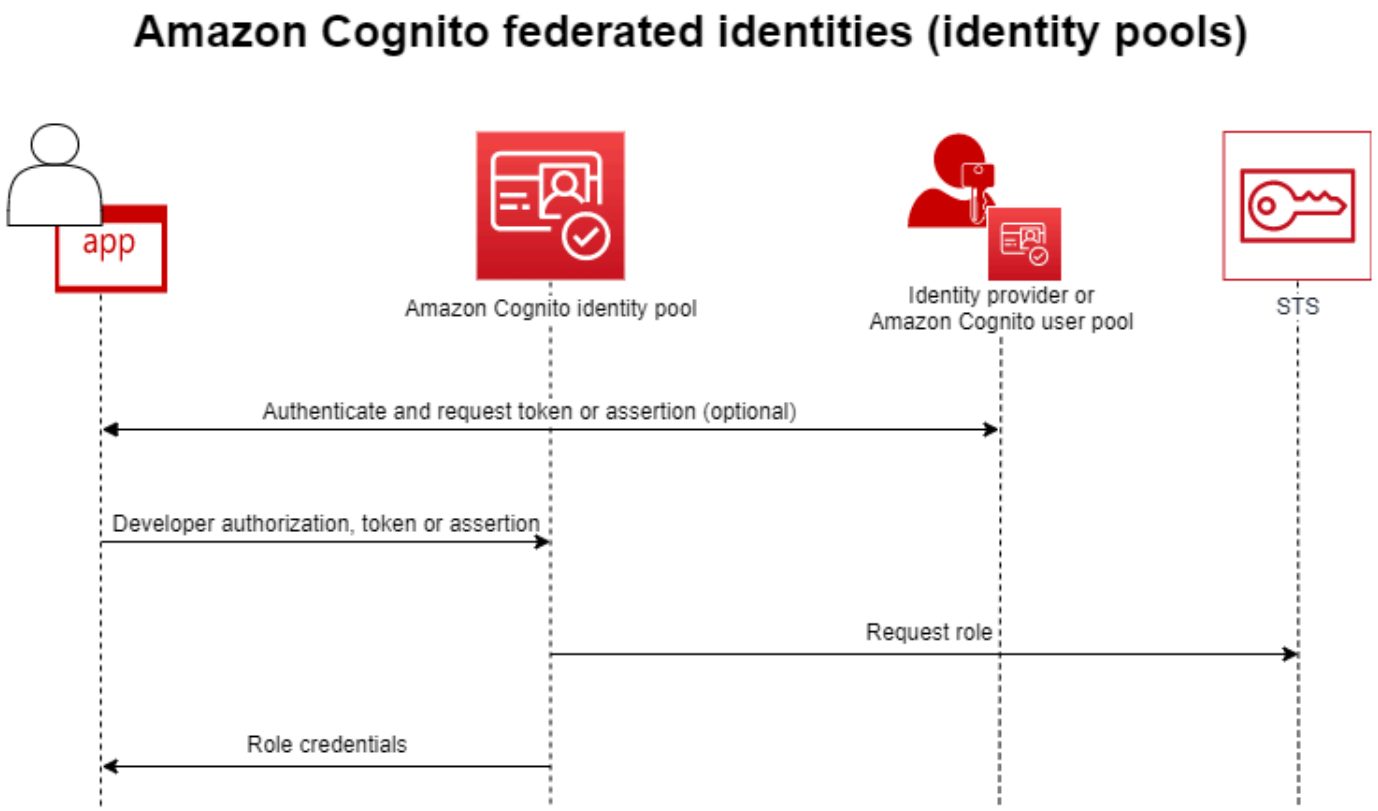
用户池



当您想要对您的应用程序或 API 进行身份验证和授权时，请创建用户池。用户池是一个用户目录，既有自助服务，也有管理员驱动的用户创建、管理和身份验证。用户池可以是独立的目录和 OIDC 身份提供者（IdP），也可以是员工身份和客户身份的第三方提供者的中间服务提供者（SP）。您可以在 SAML 2.0 和 OIDC 中为组织的工作人员身份提供单点登录 (SSO) 和带有用户池的 OIDC IdPs 。您还可以在应用程序中为组织在亚马逊、谷歌、苹果和Facebook的公共 OAuth 2.0身份存储库中的客户身份提供单点登录。有关客户身份和访问管理（CIAM）的更多信息，请参阅什么是 CIAM？

用户池不要求与身份池集成。在用户池中，您可以直接向应用程序、Web 服务器或 API 发放经过身份验证的 JSON Web 令牌 (JWTs)。

身份池



如果您想授权经过身份验证的用户或匿名用户访问您的 Amazon 资源，请设置 Amazon Cognito 身份池。身份池会为您的应用程序颁发 Amazon 凭证，以便向用户提供资源。您可以使用可信身份提供者（如用户池或 SAML 2.0 服务）对用户进行身份验证。此身份提供者还可以选择为访客用户颁发凭证。身份池使用基于角色和基于属性的访问控制来管理用户访问您的资源的授权。 Amazon

身份池不要求与用户池集成。身份池可以直接接受来自员工和使用者身份提供者的经过身份验证的声明。

将 Amazon Cognito 用户池和身份池一起使用

在本主题开头的图表中，您使用 Amazon Cognito 对用户进行身份验证，然后向他们授予对 Amazon Web Services 服务 的访问权限。

1. 您的应用程序用户通过用户池登录并收到 OAuth 2.0 个令牌。
2. 您的应用程序将用户池令牌与身份池交换为临时 Amazon 证书，您可以将这些证书与 Amazon APIs 和 Amazon Command Line Interface (Amazon CLI) 一起使用。

3. 您的应用程序会将凭证会话分配给您的用户，并授予对诸如 Amazon S3 和 Amazon DynamoDB 之 Amazon Web Services 服务 类的授权访问权限。

有关使用身份池和用户池的更多示例，请参阅常见的 Amazon Cognito 场景。

在 Amazon Cognito 中，责任共担模式的云安全义务符合 SOC 1-3、PCI DSS、ISO 27001 的要求且符合 HIPAA-BAA 的条件。你可以在 Amazon Cognito 中将云端安全设计为符合 SOC1 -3、ISO 27001 和 HIPAA-BAA，但不符合 PCI DSS。有关更多信息，请参阅范围内的Amazon 服务。另请参阅区域数据注意事项。

Amazon Cognito 的功能

用户池

Amazon Cognito 用户池是一个用户目录。利用用户池，您的用户可以通过 Amazon Cognito 登录您的 Web 或移动应用程序，也可以通过第三方 IdP 进行联合身份验证。联合用户和本地用户在您的用户池中拥有用户配置文件。

本地用户是那些已注册或您直接在用户池中创建的用户。您可以在、 Amazon SDK 或 Amazon Command Line Interface (Amazon CLI) 中 Amazon Web Services Management Console管理和自定义这些用户配置文件。

Amazon Cognito 用户池接受来自第三方的令牌和断言 IdPs，并将用户属性收集到发布给您的应用程序的 JWT 中。在 Amazon Cognito 处理与之交互的 JWTs 同时，您可以将应用程序标准化为一组 IdPs，将其声明映射到中央令牌格式。

Amazon Cognito 用户池可以是独立的 IdP。Amazon Cognito 从 OpenID Connect (OIDC) 标准中汲取灵感，生成用于身份验证和授权。 JWTs 当您登录本地用户时，用户池对这些用户具有权限。对本地用户进行身份验证时，您可以访问以下功能。

- 实现您自己的 Web 前端，此前端调用 Amazon Cognito 用户池 API 来对用户进行验证、授权和管理。
- 为用户设置多重身份验证（MFA）。Amazon Cognito 支持基于时间的一次性密码（TOTP）和短信 MFA。
- 可防止来自受恶意控制的用户账户的访问。
- 创建您自己的自定义多步骤身份验证流程。
- 在另一个目录中查找用户并将其迁移到 Amazon Cognito。

Amazon Cognito 用户池还可以充当您的应用程序的服务提供商 (SP) 和您的应用程序的 IdPs IdP 的双重角色。Amazon Cognito 用户池可以连接到 Facebook 和谷歌 IdPs 等消费者，也可以连接 Okta 和 Active Directory 联合服务 (ADFS) IdPs 等员工。

使用亚马逊 Cognito 用户池发行的 OAuth 2.0 和 OpenID Connect (OIDC) 令牌，你可以

- 在应用程序中接受 ID 令牌，该令牌可对用户进行身份验证，并提供设置用户配置文件所需的信息
- 在您的 API 中接受访问令牌，此令牌具有用于对用户的 API 调用进行授权的 OIDC 范围。
- 从 Amazon Cognito 身份池中检索 Amazon 证书。

Amazon Cognito 用户池的功能

特征	描述
OIDC 身份提供商	发放 ID 令牌对用户进行身份验证
授权服务器	发放访问令牌以授权用户访问 APIs
SAML 2.0 服务提供商	将 SAML 断言转换为 ID 和访问令牌
OIDC 依赖方	将 OIDC 令牌转换为 ID 和访问令牌
社交提供者依赖方	将苹果、Facebook、亚马逊或谷歌的 ID 令牌转换为你自己的 ID 和访问令牌
认证前端服务	使用托管登录注册、管理和验证用户
为你自己的用户界面提供 API 支持	支持通过身份验证 API 请求创建、管理和验证用户 Amazon SDKs [1]
多重身份验证	使用短信或用户的设备作为额外的身份验证因素 [1] TOTPs
安全监控和响应	抵御恶意活动和不安全的密码[1]
自定义身份验证流程	构建自己的身份验证机制，或向现有流程添加自定义步骤[2]
组	在将令牌传递给身份池时，创建用户的逻辑分组和 IAM 角色声明的层次结构

自定义代币	使用新的、修改的和禁止的声明自定义您的 ID 和访问令牌
自定义用户属性	为用户属性分配值并添加您自己的自定义属性

[1] 该功能对联合用户不可用。

[2] 联合登录用户和托管登录用户无法使用该功能。

有关用户池的更多信息，请参阅用户池入门和 Amazon Cognito 用户池 API 参考。

身份池

身份池是您分配给用户或访客并授权其接收临时 Amazon 证书的唯一标识符或身份的集合。当您以 SAML 2.0、OpenID Connect (OIDC) 或 2.0 社交身份提供商 (IdP) OAuth 的可信声明形式向身份池提供身份验证证明时，您就将您的用户与身份池中的身份关联起来。您的身份池为身份创建的令牌可以从 Amazon Security Token Service (Amazon STS) 检索临时会话证书。

为了补充经过身份验证的身份，您还可以将身份池配置为在没有 IdP 身份验证的情况下授权 Amazon 访问。您可以通过提供自定义的身份验证证明经开发人员验证的身份。您还可以向具有未经 Amazon 身份验证的访客用户授予临时证书。

使用身份池，您可以通过两种方式与自己的 IAM 策略集成 Amazon Web Services 账户。您可以同时使用这两个功能，也可以单独使用。

基于角色的访问控制

当用户将声明传递到身份池时，Amazon Cognito 会选择它请求的 IAM 角色。要根据您的需求自定义角色的权限，您可以对每个角色应用 IAM policy。例如，如果您的用户证明自己在市场营销部门工作，则他们将获得相应角色的凭证，该角色的策略是根据营销部门访问需求量身定制的。Amazon Cognito 可以请求原定设置角色、基于查询用户声明的规则来请求角色，或者基于用户在用户池中的组成员资格来请求角色。您还可以配置角色信任策略，以便 IAM 仅信任您的身份池来生成临时会话。

访问控制属性

您的身份池读取用户声明中的属性，并将它们映射到用户临时会话中的主体标签。然后，您可以配置基于 IAM 资源的 IAM policy，以基于您的身份池中携带会话标签的 IAM 主体允许或拒绝访问资源。例如，如果您的用户证明他们在市场营销部门工作，则会为他们的会话添加 Amazon STS 标签`Department: marketing`。您的 Amazon S3 存储桶允许基于 a ws: PrincipalTag 条件进行读取操作，该条件要求`Department`标签的`marketing`值为。

Amazon Cognito 身份池的功能

特征	描述
亚马逊 Cognito 用户池依赖方	使用用户池中的 ID 令牌交换来自的 Web 身份凭证 Amazon STS
SAML 2.0 服务提供商	交换来自 Web 身份凭证的 SAML 断言 Amazon STS
OIDC 依赖方	使用 OIDC 令牌兑换 Web 身份凭证 Amazon STS
社交提供者依赖方	使用来自亚马逊、Facebook、谷歌、苹果和 Twitter 的 OAuth 代币来换取来自的网络身份凭证 Amazon STS
自定义依赖方	使用 Amazon 凭证，以任何格式交换来自的 Web 身份凭证的声明 Amazon STS
未经身份验证的访问	无需身份验证即可颁发访问受 Amazon STS 限的 Web 身份凭证
基于角色的访问控制	根据身份验证用户的声明为其选择一个 IAM 角色，并将您的角色配置为仅在您的身份池环境中担任
基于属性的访问控制	将声明转换为 Amazon STS 临时会话的委托人标签，并使用 IAM 策略根据委托人标签筛选资源访问权限

有关身份池的更多信息，请参阅Amazon Cognito 身份池入门和 Amazon Cognito 身份池 API 参考。

Amazon Cognito 用户池和身份池的比较

特征	描述	用户池	身份池

OIDC 身份提供商	发放 OIDC ID 令牌以对应用程序用户进行身份验证	✓	
用户目录	存储用户配置文件以进行身份验证	✓	
授权 API 访问权限	发放访问令牌以授权用户访问接受 OAuth 范围的资源 APIs（包括用户个人资料自助服务 API 操作）、数据库和其他资源	✓	
IAM 网络身份授权	生成可用于交换临时 Amazon 凭证 Amazon STS 的令牌		✓
SAML 2.0 服务提供商和 OIDC 身份提供商	根据 SAML 2.0 身份提供商的声明发行自定义 OIDC 令牌	✓	
OIDC 依赖方和 OIDC 身份提供商	根据OIDC身份提供商的声明发行定制的OIDC令牌	✓	
OAuth 2.0 依赖方和 OIDC 身份提供商	根据苹果和谷歌等OAuth 2.0社交提供商的范围发行定制的OIDC代币	✓	
SAML 2.0 服务提供商和凭证代理	根据 SAML 2.0 身份提供商的声明颁发临时 Amazon 证书		✓
OIDC 信赖方和凭证代理	根据 OIDC Amazon 身份提供商的声明颁发临时证书		✓

社交提供者依赖方和证书经纪人	根据开发者应用程序向 Apple 和 Google 等社交提供商提供的 JSON 网络令牌颁发临时 Amazon 证书		✓
Amazon Cognito 用户池信赖方和凭证代理	根据来自 Amazon Cognito 用户池的 JSON 网络令牌颁发临时 Amazon 证书		✓
自定义信赖方和凭证代理	向由开发者 IAM Amazon 证书授权的任意身份颁发临时证书		✓
认证前端服务	使用托管登录注册、管理和验证用户	✓	
API 支持你自己的身份验证界面	通过支持的 API 请求创建、管理和验证用户 Amazon SDKs [1]	✓	
MFA	使用短信或用户的设备作为额外的身份验证因素[1] TOTPs	✓	
安全监控和响应	防范恶意活动和不安全的密码[1]	✓	
自定义身份验证流程	构建自己的身份验证机制，或在现有流程中添加自定义步骤[1]	✓	
用户组	在将令牌传递给身份池时，创建用户的逻辑分组和 IAM 角色声明的层次结构	✓	

自定义代币	使用新的、修改的和禁止的声明和范围自定义您的 ID 和访问令牌	✓	
Amazon WAF 网页 ACLs	通过以下方式监控和控制对身份验证前端的请求 Amazon WAF	✓	
自定义用户属性	为用户属性分配值并添加您自己的自定义属性	✓	
未经身份验证的访问	无需身份验证即可颁发访问受 Amazon STS 限的 Web 身份凭证		✓
基于角色的访问控制	根据身份验证用户的声明为其选择 IAM 角色，然后配置您的角色信任以限制 Web 身份用户的访问权限		✓
基于属性的访问控制	将用户声明转换为 Amazon STS 临时会话的委托人标签，并使用 IAM 策略根据委托人标签筛选资源访问权限		✓

[1] 该功能不适用于联合用户。

Amazon Cognito 入门

有关用户池应用程序的示例，请参阅用户池入门。

有关身份池的介绍，请参阅Amazon Cognito 身份池入门。

有关用户池和身份池的引导式设置体验的链接，请参阅Amazon Cognito 的引导式设置选项。

要开始使用 S Amazon DK，请参阅Amazon 开发者工具。有关特定于 Amazon Cognito 的开发者资源，请参阅亚马逊 Cog nito 开发者资源。

您需要 Amazon Web Services 账户才能使用 Amazon Cognito。有关更多信息，请参阅 入门 Amazon。

区域可用性

亚马逊 Cognito 已在全球多个 Amazon 地区上市。在每个区域中，Amazon Cognito 分布在多个可用区内。这些可用区的物理位置是相互隔离的，但可通过私有、低延迟、高吞吐量和高度冗余的网络连接联合在一起。这些可用区 Amazon 能够为包括 Amazon Cognito 在内的服务提供非常高的可用性和冗余性，同时还可以最大限度地减少延迟。

要查看 Amazon Cognito 目前是否在任何版本中可用 Amazon Web Services 区域，请参阅按地区划分的Amazon 服务。

要了解有关区域 API 服务端点的信息，请参阅 Amazon Web Services 一般参考 中的 Amazon 区域和端点。

要详细了解每个区域中可用的可用区数量，请参阅 Amazon 全球基础设施。

Amazon Cognito 的定价

有关 Amazon Cognito 定价的信息，请参阅 Amazon Cognito 定价。

常见 Amazon Cognito 术语和概念

Amazon Cognito 为 Web 和移动应用程序提供凭证。它借鉴身份和访问管理中的常见术语并在此基础上构建。有许多关于通用身份和访问管理术语的指南可供参考。部分示例包括：

- 知识 IDPro 体系中的@@ 术语
- Amazon 身份服务
- 来自 NIST CSRC 的术语表

以下列表说明了 Amazon Cognito 独有的术语或在 Amazon Cognito 中具有特定上下文的术语。

主题

- 常规
- 用户池
- 身份池

常规

此列表中的术语并非特定于 Amazon Cognito，而是在身份和访问管理从业人员中得到广泛认可的术语。以下并不是术语的详尽列表，而是关于这些术语在本指南中的特定 Amazon Cognito 上下文中的解释。

访问令牌

一个 JSON 网络令牌 (JWT)，其中包含有关实体访问信息系统的授权的信息。

应用程序、应用程序

通常是一个移动应用程序。在本指南中，应用通常是连接到 Amazon Cognito 的 Web 应用程序或移动应用程序的简写。

基于属性的访问控制（ABAC）

一种模型，在这种模型中，应用程序根据用户的属性（例如其职位或部门）来确定对资源的访问权限。实施 ABAC 的 Amazon Cognito 工具包括用户池中的 ID 令牌和身份池中的主体标签。

身份验证

为访问信息系统而建立真实身份的过程。Amazon Cognito 接受来自第三方身份提供商的身份验证证明，同时也为软件应用程序提供身份验证。

授权

向资源授予权限的过程。用户池访问令牌包含应用程序可用于允许用户和系统访问资源的信息。

授权服务器

生成 JSON 网络令牌的 OAuth 或 OpenID Connect (OIDC) 系统。Amazon Cognito 用户池托管授权服务器是用户池中两种身份验证和授权方法的授权服务器组件。用户池还支持 SDK 身份验证中的 API 质询-响应流程。

机密应用程序、服务器端应用程序

用户远程连接的应用程序，其代码位于应用程序服务器上，并且可以访问密钥。这通常是 Web 应用程序。

身份提供者（IdP）

一个存储和验证用户身份的服务。Amazon Cognito 可以向外部提供者请求身份验证，并可以成为应用程序的 IdP。

JSON Web 令牌（JWT）

一个 JSON 格式的文档，其中包含有关经过身份验证的用户的声明。ID 令牌对用户进行身份验证，访问令牌授权用户，刷新令牌更新凭证。Amazon Cognito从外部提供商那里接收令牌，并向应用程序发放令牌，或者. Amazon STS

Machine-to-machine (M2M) 授权

授权向 non-user-interactive机器实体（例如 Web 服务器应用程序层）的 API 端点发出请求的过程。用户池在客户端凭证授予中提供 M2M 授权，访问令牌中的范围为 OAuth 2.0。

多重身份验证 (MFA)

要求用户在提供用户名和密码后提供额外的身份验证。Amazon Cognito 用户池具有本地用户的 MFA 特征。

OAuth 2.0（社交）提供商

用户池或身份池的 IdP，用于提供 JWT 访问令牌和刷新令牌。在用户进行身份验证后，Amazon Cognito 用户池自动与社交提供者进行交互。

OpenID Connect（OIDC）提供者

用户池或身份池的 IdP，用于扩展OAuth规范以提供 ID 令牌。在用户进行身份验证后，Amazon Cognito 用户池自动与 OIDC 提供者进行交互。

Passkey， WebAuthn

一种身份验证形式，用户设备上的加密密钥或密钥提供其身份验证证明。用户验证硬件或软件身份验证器中是否存在生物识别或 PIN 码机制。Passkeys 可以抵御网络钓鱼，并绑定到特定的网站/应用程序，从而提供安全的无密码体验。Amazon Cognito 用户池支持使用密钥登录。

无密码

一种用户无需输入密码的身份验证形式。无密码登录的方法包括发送到电子邮件地址和电话号码的一次性密码 (OTPs) 以及密钥。Amazon Cognito 用户池支持使用和密钥登录。 OTPs

公共应用程序

设备上的一个独立应用程序，其代码存储在本地，并且无法访问密钥。这通常是移动应用程序。

资源服务器

具有访问控制的 API。Amazon Cognito 用户池还使用资源服务器来描述组件，该组件定义了用于与 API 进行交互的配置。

基于角色的访问控制 (RBAC)

一个根据用户的职能名称授予访问权限的模型。Amazon Cognito 身份池通过区分 IAM 角色来实施 RBAC。

服务提供者（SP）、依赖方（RP）

依赖 IdP 来断言用户值得信赖的应用程序。Amazon Cognito 充当外部的 SP，对于基于应用程序的 IdP IdPs，则充当 IdP。 SPs

SAML 提供商

用户池或身份池的 IdP，用于生成经过数字签名的断言文档，您的用户会将该文档传递给 Amazon Cognito。

通用唯一标识符（UUID）

应用于对象的 128 位标签。Amazon Cognito UUIDs 在每个用户池或身份池中都是唯一的，但不符合特定的 UUID 格式。

用户目录

一组用户及其属性，可以向其他系统提供该信息。Amazon Cognito 用户池是用户目录，也是用于整合外部用户目录中用户的工具。

用户池

当您在本指南中看到以下列表中的术语时，它们指的是用户池的特定特征或配置。

自适应身份验证

一项高级安全特征，用于检测潜在的恶意活动并为用户配置文件增加额外的安全保护。

高级安全特征

一个可选组件，用于添加用户安全工具。

应用程序客户端

一个组件，用于将用户池的设置定义为一个应用程序的 IdP。

回调网址、重定向 URI、返回网址

应用程序客户端中的设置和对用户池授权服务器的请求中的参数。回调 URL 是您的应用程序中经过身份验证的用户的初始目的地。

基于选择的身份验证

一种使用用户池的 API 身份验证形式，其中每个用户都有一组可供他们选择的登录选项。他们的选择可能包括带或不带 MFA 的用户名和密码、密钥登录或使用电子邮件或短信一次性密码进行无密码登录。您的应用程序可以通过请求身份验证选项列表或声明首选选项来决定用户的选择流程。

请与基于客户端的身份验证进行比较。

基于客户端的身份验证

一种使用用户池 API 和应用程序后端构建的身份验证形式 Amazon SDKs。在声明式身份验证中，您的应用程序独立确定用户应执行的登录类型，并预先请求该类型。

请与基于选择的身份验证进行比较。

已泄露的凭证

一项高级安全特征，用于检测攻击者可能知道的用户密码，并为用户配置文件增加额外的安全保护。

确认

一个过程，用于确定已满足先决条件，可允许新用户登录。通常通过电子邮件地址或电话号码验证来完成确认。

自定义身份验证

使用 Lambda 触发器的身份验证过程的一个扩展，定义了额外的用户质询和响应。

设备身份验证

一个身份验证过程，将 MFA 替换为使用可信设备 ID 的登录。

域、用户池域

托管您的托管登录页面的网域 Amazon。您可以在自己拥有的域中设置 DNS，也可以在 Amazon 拥有的域中使用可识别的子域名前缀。

必备套餐

包含用户池最新发展的功能计划。Essentials 计划在 Plus 计划中不包括自动学习安全功能。

外部提供者、第三方提供者

与用户池存在信任关系的 IdP。用户池充当外部提供商和您的应用程序之间的中间实体，使用 SAML 2.0、OIDC 和社交提供商管理身份验证流程。用户池将外部提供商的身份验证结果整合到单个 IdP 中，以便您的应用程序可以使用单个 OIDC relying-party 库处理多个用户。

功能计划

您可以为用户池选择的一组功能。功能套餐的 Amazon 账单费用各不相同。新用户池默认为基本套餐。

目前的计划

- 精简版套餐
- 必备套餐
- Plus 计划

联邦用户、外部用户

用户池中由外部提供者进行身份验证的用户。

托管用户界面（经典）、托管用户界面页面

您的用户池域上身份验证前端、信赖方和身份提供商服务的早期版本。托管用户界面具有一组基本功能和简化的外观。您可以通过上传徽标图像文件和带有预先确定的 CSS 样式集的文件来应用托管 UI 品牌。与托管登录进行比较。

Lambda 触发器

其中一种函数 Amazon Lambda ，用户池可以在用户身份验证过程的关键时刻自动调用。您可以使用 Lambda 触发器自定义身份验证结果。

本地用户

用户池用户目录中的用户配置文件，它不是通过外部提供者进行身份验证而创建。

关联用户

来自外部提供者的用户，其身份与本地用户合并。

精简版套餐

包含最初通过用户池启动的功能的功能计划。精简版计划不包括基本计划中的新功能或增强版计划中的自动学习安全功能。

托管授权服务器、托管 UI 授权服务器、授权服务器

托管登录的一个组件，用于在用户池域上托管 IdPs 与之交互的服务和应用程序。托管用户界面与托管登录的不同之处在于它提供的用户交互功能，但具有相同的授权服务器功能。

托管登录、托管登录页面

用户池域上的一组网页，用于托管用户身份验证服务。这些服务包括作为 IdP IdPs、第三方依赖方以及用户交互式身份验证 UI 服务器运行的功能。当您为用户池设置域名时，Amazon Cognito 会将所有托管登录页面联机。

您的应用程序会导入 OIDC 库，这些库可以调用用户的浏览器，并将他们定向到托管登录界面进行注册、登录、密码管理和其他身份验证操作。身份验证后，OIDC 库可以处理身份验证请求的结果。

托管登录身份验证

使用用户池网域上的服务登录，通过用户交互式浏览器页面或 HTTPS API 请求完成。应用程序使用 OpenID Connect (OIDC) 库处理托管登录身份验证。此过程包括向外部提供商登录、使用交互式托管登录页面进行本地用户登录以及 M2M 授权。使用经典托管 UI 进行身份验证也属于该术语。

请与 Amazon SDK 身份验证进行比较。

Plus 计划

该功能计划包含用户池中的最新开发和高级安全功能。

SDK 身份验证、 Amazon SDK 身份验证

一组身份验证和授权 API 操作，您可以使用 Amazon SDK 将其添加到应用程序后端。此身份验证模型需要您自己的自定义登录机制。该 API 可以登录本地用户和关联用户。

请与托管登录身份验证进行比较。

威胁防护

在用户池中，威胁防护是指旨在缓解对您的身份验证和授权机制的威胁的技术。自适应身份验证、凭据泄露检测和 IP 地址屏蔽名单属于威胁防护类别。

令牌自定义

令牌生成前 Lambda 触发器的结果，该触发器在运行时修改用户的 ID 令牌或访问令牌。

用户池、Amazon Cognito 身份提供者、**`cognito-idp`**、Amazon Cognito 用户池

为使用 OID IdPs C 的应用程序提供身份验证和授权服务的 Amazon 资源。

验证

确认用户拥有电子邮件地址或电话号码的过程。用户池向输入新电子邮件地址或电话号码的用户发送代码。当他们向 Amazon Cognito 提交代码时，他们会验证自己对消息目的地的所有权，并且可以从用户池中接收其他消息。另请参阅确认。

用户配置文件、用户账户

用户目录中的用户条目。所有用户（包括来自第三方 IdPs的用户）在其用户池中都有个人资料。

身份池

当您在本指南中看到以下列表中的术语时，它们指的是身份池的特定特征或配置。

访问控制属性

身份池中基于属性的访问控制的一种实施。身份池将用户属性作为标签应用于用户凭证。

基本（经典）身份验证

一个身份验证过程，您可以在这个过程中自定义对用户凭证的请求。

已经过开发人员验证的身份

一个身份验证过程，在这个过程中使用开发人员凭证授权身份池用户凭证。

开发人员凭证

身份池管理员的 IAM API 密钥。

增强型身份验证

一个身份验证流程，根据您在身份池中定义的逻辑选择 IAM 角色并应用主体标签。

身份

一个 UUID，用于将应用程序用户及其用户凭证链接到与身份池存在信任关系的外部用户目录中的配置文件。

身份池、Amazon Cognito 联合身份、Amazon Cognito 身份、**`cognito-identity`**

一种 Amazon 资源，为使用临时 Amazon 证书的应用程序提供身份验证和授权服务。

未经验证的 身份

尚未使用身份池 IdP 登录的用户。您可以允许用户在进行身份验证之前为单个 IAM 角色生成有限的用户凭证。

用户凭证

用户在身份池身份验证后收到的临时 Amazon API 密钥。

入门 Amazon

在开始使用 Amazon Cognito 之前，请为自己设置一些必需 Amazon 的资源。如果您已经可以登录 Amazon Web Services 账户，则可以跳过此部分。如果您正在寻找有关注册和使用 Amazon 凭据登录的信息，请继续阅读。获得具有足够 Amazon Identity and Access Management (IAM) 权限的证书后，就可以开始使用用户池和身份池了。

注册获取 Amazon Web Services 账户

如果您没有 Amazon Web Services 账户，请完成以下步骤来创建一个。

要注册 Amazon Web Services 账户

1. 打开https://portal.aws.amazon.com/billing/注册。
2. 按照屏幕上的说明操作。

   在注册时，将接到电话，要求使用电话键盘输入一个验证码。

   当您注册时 Amazon Web Services 账户，就会创建Amazon Web Services 账户根用户一个。根用户有权访问该账户中的所有 Amazon Web Services 服务 和资源。作为最佳安全实践，请为用户分配管理访问权限，并且只使用根用户来执行需要根用户访问权限的任务。

Amazon 注册过程完成后会向您发送一封确认电子邮件。您可以随时前往 https://aws.amazon.com/并选择 “我的账户”，查看您当前的账户活动并管理您的账户。

保护 IAM 用户

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

要允许其他用户访问您的 Amazon Web Services 账户 资源，请创建 IAM 用户。为了保护您的 IAM 用户，请启用 MFA 并仅向 IAM 用户授予执行任务所需的权限。

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

- 在你的 IAM 用户中创建 Amazon Web Services 账户

- 适用于 Amazon 资源的访问权限管理
- 基于 IAM 身份的策略示例

用户池入门

您的应用程序需要身份验证和访问控制。你可以在 OpenID Connect (OIDC) 框架内进行单点登录 (SSO)。Amazon Cognito 提供了一些工具，用于使用 Amazon 软件开发工具包处理应用程序后端的身份验证逻辑，以及调用客户端中的浏览器来访问托管授权服务器。

Amazon Cognito 控制台将指导您从首选应用程序框架的角度创建用户池。然后，您可以继续添加功能，例如通过外部社交或 SAML 2.0 身份提供商进行联合登录（ ）IdPs。Amazon Cognito 控制台中的应用程序模型依赖于在您的项目中添加 OIDC 库并调用浏览器。

当您计划扩展特征集并整合 Amazon Cognito 的更多组件时，请阅读 Amazon Cognito 用户池章节，以获取有关用户池所能提供的所有特征的详细描述。

本章和 Amazon Cognito 控制台中的示例演示了应用程序资源与 Amazon Cognito 用户池的基本集成。稍后，您可以调整用户池以使用更多可用的选项。然后，您可以更新您的应用程序以采用新功能并与之交互 IdPs。

如果您不想使用托管登录页面，则可以使用 Amazon SDK 或 Amazon Amplify创建带有自定义身份验证接口的应用程序。您以这种方式构建的应用程序与用户池 API 交互，并且仅适用于对本地用户进行身份验证。要继续了解此身份验证模型，请访问其他应用程序选项。

主题

- 在 Amazon Cognito 控制台中创建新应用程序
- 其他应用程序选项
- 向您的用户池添加更多特征和安全选项

在 Amazon Cognito 控制台中创建新应用程序

用户池向软件应用程序添加身份验证选项。要获得最简单的入门体验，请进入 Amazon Cognito 控制台并按照那里的说明进行操作。那里的创建过程不仅可以指导您设置用户池资源，还可以指导您设置应用程序的初始部分。

准备好开始时，请导航到 Amazon Cognito 控制台，然后选择按钮以创建新的用户池。设置过程将指导您完成配置和编程语言选项。

有关身份验证概念的其他资源

- 使用 Amazon Cognito 用户池进行身份验证

- 了解 API、OIDC 和托管登录页面身份验证
- 身份验证如何与 Amazon Cognito 配合使用
- 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成

为您的应用程序创建 Amazon Cognito 资源

1. 导航到 Amazon Cognito 控制台。要向您的 IAM 委托人分配权限，以便他们可以创建和管理 Amazon Cognito 资源，请参阅。Amazon 亚马逊 Cognito 的托管策略该`AmazonCognitoPowerUser`策略足以创建用户池。
2. 从 “用户池” 菜单中选择 “创建用户池”，或者选择 “在不到五分钟的时间内免费开始使用”。
3. 在 “定义您的应用程序” 下，选择最适合您要为其创建身份验证和授权服务的应用程序场景的应用程序类型。
4. 在命名您的应用程序中，输入描述性名称或继续使用默认名称。
5. 您必须在配置选项下做出一些基本选择，这些选项支持在创建用户池后无法更改的设置。

   a. 在 “登录标识符选项” 下，告诉我们您希望在用户登录时如何识别他们。您可以首选用户生成的用户名、电子邮件地址或电话号码。您也可以允许多个选项的组合。Amazon Cognito 接受您在托管登录表单的用户名字段中配置的选项。

   b. 在 “注册必填属性” 下，告诉我们用户注册新账户时您要收集哪些用户信息。在托管登录页面中，Amazon Cognito 会显示所有必需属性的提示。

      登录标识符的选项会影响您的所需属性。用户名需要为每个用户提供电子邮件或电话属性，以便他们可以在电子邮件或短信中收到密码重置代码。电子邮件需要电子邮件属性，电话号码需要电话号码属性。
6. 在 “添加返回 URL” 下，输入用户完成身份验证后的应用程序重定向路径。此位置应该是您的应用程序中使用 OpenID Connect (OIDC) 库来处理用户身份验证结果的路由。测试应用程序的返回 URL 的示例是`https://localhost:3000/callback`。在 Amazon Cognito 控制台的 nodeJS 应用程序示例，此路由使用 openid-client 来收集访问令牌并将其兑换为用户信息。创建资源后，您将能够浏览开发平台的示例。
7. 选择 “创建您的应用程序”。Amazon Cognito 使用您的应用程序类型的默认设置创建用户池和应用程序客户端。创建初始资源后，您可以配置其他选项，例如外部身份提供商和多因素身份验证 (MFA)。
8. 在设置您的应用程序页面上，您可以立即获取应用程序的代码示例。要浏览您的新用户池，请向下滚动并选择 “转至概览”。

9. 要在同一个用户池中添加更多应用程序，请导航到应用程序客户端菜单并添加新的应用程序客户端。这将重复以应用程序为中心的创建过程，但仅向现有用户池中添加新的应用程序客户端。

使用此过程创建用户池和一个或多个应用程序客户端后，您可以开始使用托管登录测试身份验证操作。这些快速入门选项可供公众自行注册。我们建议您使用控制台流程创建测试环境，然后将最终设计移至生产环境。花点时间熟悉 Amazon Cognito 的功能。然后，要转移到生产工作负载，请制作自定义配置，并使用自动化工具（如 Amazon CloudFormation 和）进行部署 Amazon Cloud Development Kit (Amazon CDK)。

Amazon Cognito 在此过程中会进行一些您无法撤消的默认配置。有关您无法更改的用户池设置以及可以在控制台中选择的选项的更多信息，请参阅更新用户池和应用程序客户端配置。

设置	效果	如何改变	更多信息
客户端密钥	需要在身份验证请求中使用客户端密钥哈希。	使用传统 Web 应用程序或应用程序配置文件创建新的Machine-to-machine 应用程序客户端。	特定于应用程序的应用程序客户端设置
首选用户名	用户池不接受该`preferred_username` 属性作为别名。	使用 Amazon SDK 以编程方式创建用户池。	自定义登录属性
区分大小写	用户池用户名不区分大小写，例如`JohnD`被视为与用户相同的用户。`johnd`	使用 Amazon SDK 以编程方式创建用户池。	用户池区分大小写

其他应用程序选项

您可能有一个想要与 Amazon Cognito 身份验证集成的现有应用程序用户界面。您甚至可能拥有自己的现有身份验证页面，其目录设置功能不如 Amazon Cognito 用户池那么实用。您可以使用适用于各种编程语言的 Amazon Cognito 集成向此类应用程序添加或替换身份验证组件。 Amazon SDKs 以下为一些示例。

如果您在 Amazon Cognito 控制台中为此目的创建用户池，则可能没有必要拥有托管交互式登录页面和 OpenID Connect (OIDC) 服务的用户池域。在控制台中创建用户池的过程会自动为您生成一个域。您可以从用户池的 “域” 选项卡中删除此域。其他选项包括通过编程方式为您的应用程序创建 Amazon Cognito 资源， Amazon SDKs 并在 CLI 中使用自动设置选项。 Amazon Amplify 有关更多信息，请参阅 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成。

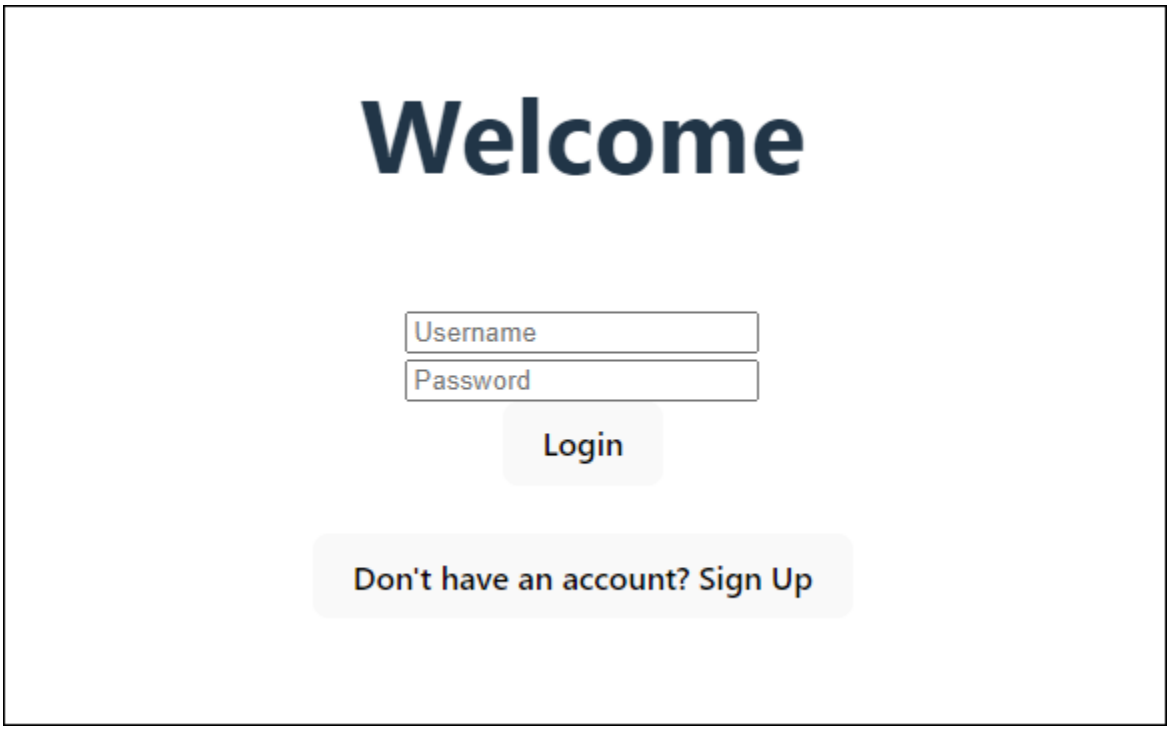
主题

- 设置一个示例 React 单页应用程序
- 使用 Flutter 设置示例 Android 应用程序

设置一个示例 React 单页应用程序

在本教程中，您将创建一个 React 单页应用程序，您可以在该应用程序中测试用户注册、确认和登录。React 是一个 JavaScript基于 Web 和移动应用程序的库，侧重于用户界面 (UI)。此示例应用程序演示了 Amazon Cognito 用户池的一些基本功能。如果您已经有使用 React 开发 Web 应用程序的经验，请从中下载示例应用程序 GitHub。

以下屏幕截图是您要创建的应用程序中的初始身份验证页面。



要设置此应用程序，您的用户池必须满足以下要求：

- 用户可以使用自己的电子邮件地址登录。Cognito 用户池登录选项：电子邮件。

- 用户名不区分大小写。用户名要求：未选择将用户名设为区分大小写。
- 多重身份验证（MFA）不是必需。MFA 强制执行：可选 MFA。
- 您的用户池通过电子邮件消息验证用于确认用户配置文件的属性。要验证的属性：发送电子邮件消息，验证电子邮件地址。
- 电子邮件是唯一必需的属性。必需属性：电子邮件。
- 用户可以在您的用户池中自行注册。自行注册：选中启用自行注册。
- 您的初始应用程序客户端是允许使用用户名和密码登录的公共客户端。应用程序类型：公共客户端，身份验证流程：`ALLOW_USER_PASSWORD_AUTH`。

创建 应用程序

要构建此应用程序，您必须设置开发人员环境。开发人员环境要求是：

1. 已安装并更新 Node.js。
2. 已安装节点包管理器（npm）并更新至版本 10.2.3 及更高版本。
3. 可以在 Web 浏览器中通过 TCP 端口 5173 访问该环境。

创建示例 React Web 应用程序

1. 登录开发人员环境并导航到应用程序的父目录。

```
cd ~/path/to/project/folder/
```

2. 创建新 React 服务。

```
npm create vite@latest frontend-client -- --template react-ts
```

3. 从上的 Amazon 代码示例存储库中克隆`cognito-developer-guide-react-example`项目文件夹 GitHub。

```
cd ~/some/other/path
```

```
git clone https://github.com/awsdocs/aws-doc-sdk-examples.git
```

```
cp -r ./aws-doc-sdk-examples/javascriptv3/example_code/cognito-identity-provider/
scenarios/cognito-developer-guide-react-example/frontend-client ~/path/to/project/
folder/
```

4. 导航到项目中的 `src` 目录。

```
cd ~/path/to/project/folder/frontend-client/src
```

5. 编辑 `config.json` 并替换以下值：

   a. `YOUR_AWS_REGION`用 Amazon Web Services 区域 代码替换。例如：`us-east-1`。

   b. 将 `YOUR_COGNITO_USER_POOL_ID` 替换为您指定用于测试的用户池的 ID。例如：`us-east-1_EXAMPLE`。用户池必须位于您在 Amazon Web Services 区域 上一步中输入的。

   c. 将 `YOUR_COGNITO_APP_CLIENT_ID` 替换为您指定用于测试的应用程序客户端的 ID。例如：`1example23456789`。应用程序客户端必须位于上一步的用户池中。

6. 如果要从 `localhost` 以外的 IP 访问示例应用程序，请编辑 `package.json` 并将行 `"dev": "vite",` 更改为 `"dev": "vite --host 0.0.0.0",`。

7. 安装应用程序。

```
npm install
```

8. 启动应用程序。

```
npm run dev
```

9. 在 Web 浏览器中通过 `http://localhost:5173` 或 `http://[IP address]:5173` 访问应用程序。

10. 使用有效的电子邮件地址注册新用户。

11. 从电子邮件消息中检索确认码。在应用程序中输入确认码。

12. 使用用户名和密码登录。

使用 Amazon Lightsail 创建 React 开发者环境

开始使用此应用程序的一种快速方法是使用 Amazon Lightsail 创建虚拟云服务器。

借助 Lightsail，您可以快速创建一个小型服务器实例，该实例已预先配置了此示例应用程序的先决条件。您可以使用基于浏览器的客户端通过 SSH 连接到您的实例，并通过公有或私有 IP 地址连接到 Web 服务器。

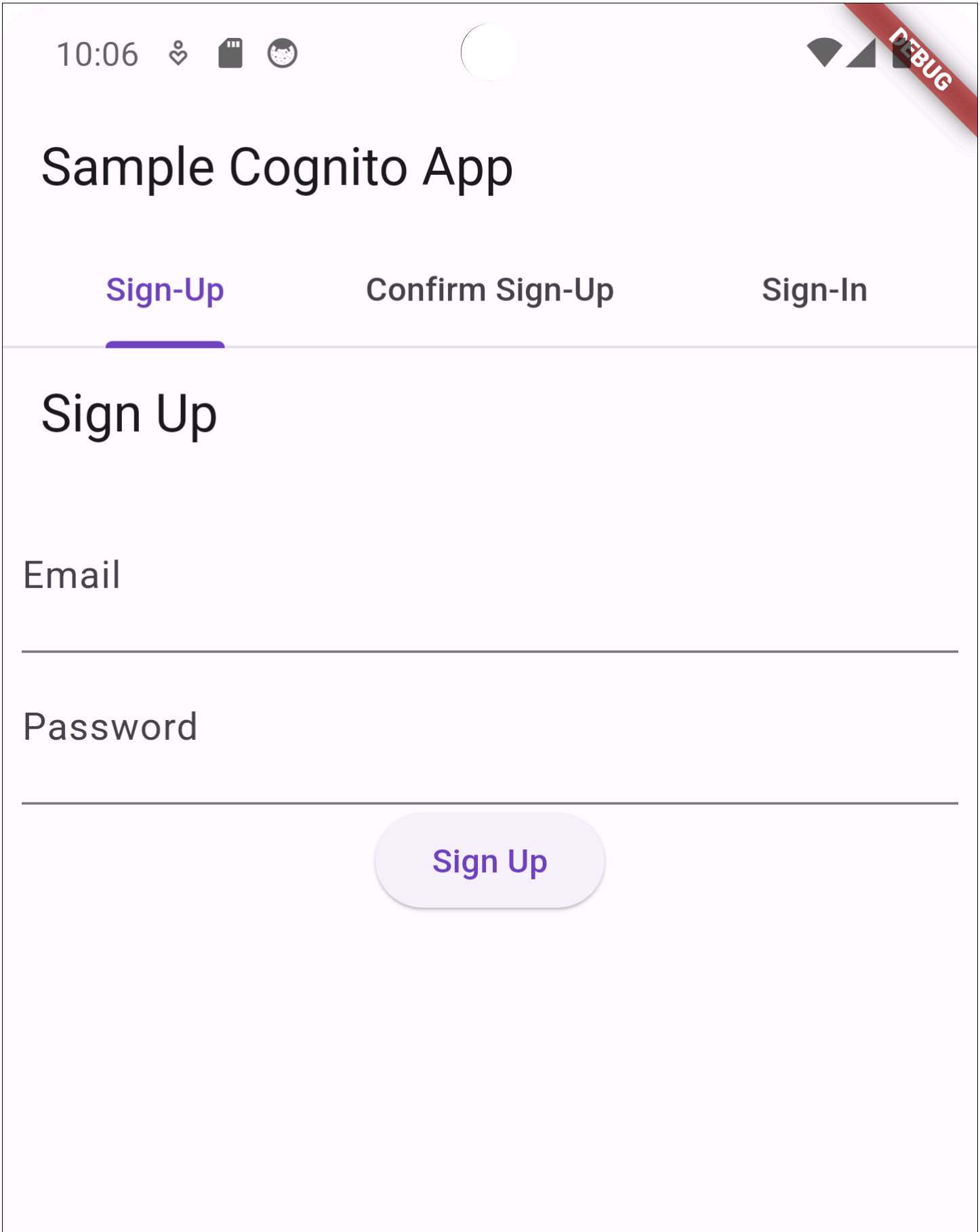
为此示例应用程序创建 Lightsail 实例

1. 前往 Lightsail 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择创建实例。
3. 在选择平台中，选择 Linux/Unix。
4. 在选择蓝图中，选择 Node.js。
5. 在标识您的实例下，为您的开发环境指定一个友好名称。
6. 选择创建实例。
7. Lightsail 创建您的实例后，将其选中，然后从 “连接” 选项卡中选择 “使用 SSH 连接”。
8. 此时会在浏览器窗口中打开 SSH 会话。运行 `node -v` 和 `npm -v` 来确认您的实例已配置了 Node.js 且最低 npm 版本为 10.2.3。
9. 继续配置 React 应用程序。

使用 Flutter 设置示例 Android 应用程序

在本教程中，您将在 Android Studio 中创建一个移动应用程序，您可以在该应用程序中模拟设备并测试用户注册、确认和登录。此示例应用程序在 Flutter 中创建一个适合 Android 的基本 Amazon Cognito 用户池移动客户端。如果您已经有使用 Flutter 开发移动应用程序的经验，请从 GitHub中下载示例应用程序。

以下屏幕截图显示在虚拟 Android 设备上运行的应用程序。


10:06
DEBUG
Sample Cognito App
Sign-Up
Confirm Sign-Up
Sign-In
Sign Up
Email
Password
Sign Up

要设置此应用程序，您的用户池必须满足以下要求：

- 用户可以使用自己的电子邮件地址登录。Cognito 用户池登录选项：电子邮件。
- 用户名不区分大小写。用户名要求：未选择将用户名设为区分大小写。
- 多重身份验证（MFA）不是必需。MFA 强制执行：可选 MFA。
- 您的用户池通过电子邮件消息验证用于确认用户配置文件的属性。要验证的属性：发送电子邮件消息，验证电子邮件地址。
- 电子邮件是唯一必需的属性。必需属性：电子邮件。
- 用户可以在您的用户池中自行注册。自行注册：选中启用自行注册。
- 您的初始应用程序客户端是允许使用用户名和密码登录的公共客户端。应用程序类型：公共客户端，身份验证流程：`ALLOW_USER_PASSWORD_AUTH`。

创建 应用程序

创建示例 Android 应用程序

1. 安装 Android studio 和命令行工具。
2. 在 Android Studio 中，安装 Flutter 插件。
3. 根据这个示例应用程序中 `cognito_flutter_mobile_app` 目录的内容创建一个新的 Android Studio 项目。
   - 编辑`assets/config.json`<<YOUR USER POOL ID>>和<< YOUR CLIENT ID>>替换 IDs 为用户池和应用程序客户端的。
4. 安装 Flutter。
   a. 将 Flutter 添加到 PATH 变量中。
   b. 使用以下命令接受许可证。

      ```
      flutter doctor --android-licenses
      ```

   c. 验证您的 Flutter 环境并安装所有缺失的组件。

      ```
      flutter doctor
      ```

      - 如果缺少任何组件，请运行 `flutter doctor -v` 以了解如何修复问题。

   d. 切换到新 Flutter 项目的目录并安装依赖项。

      - 运行 `flutter pub add amazon_cognito_identity_dart_2`。

   e. 运行 `flutter pub add flutter_secure_storage`。
5. 创建虚拟 Android 设备。
   1. 在 Android Studio GUI 中，使用设备管理器创建新设备。
   2. 在 CLI 中，运行 `flutter emulators --create --name android-device`。
6. 启动您的虚拟 Android 设备。
   1. 在 Android Studio GUI 中，选择虚拟设备旁边的启动

      图标。
   2. 在 CLI 中，运行 `flutter emulators --launch android-device`。
7. 在虚拟设备上启动应用程序。
   1. 在 Android Studio GUI 中，选择部署

      图标。
   2. 在 CLI 中，运行 `flutter run`。
8. 在 Android Studio 中导航到正在运行的虚拟设备。
9. 使用有效的电子邮件地址注册新用户。
10. 从电子邮件消息中检索确认码。在应用程序中输入确认码。
11. 使用用户名和密码登录。

向您的用户池添加更多特征和安全选项

按照教程完成示例应用程序后，可以扩大用户池实施的范围。或者，如果您没有创建测试应用程序，请根据自己的喜好创建一个新的用户池。您可以为其他应用程序自定义用户池功能，也可以添加外部身份提供商。在计划将 Amazon Cognito 用户池移至生产应用程序时，您可以评估其他示例和教程。

如果您的下一个优先事项是检查并应用用户池中的应用程序安全选项，请参阅Amazon Cognito 用户池安全最佳实践。

Amazon Cognito 的功能计划可在您选择加入更高级别时添加功能和安全选项。您可以从 Lite 计划开始，在 E ssentials 计划中添加高级身份验证和授权选项，然后使用 Plus 计划添加自动推理安全护栏。有关更多信息，请参阅 用户池功能计划。

以下是一些其他 Amazon Cognito 用户池特征：

- 将品牌应用于托管登录页面
- 向用户池添加 MFA
- 具有威胁防护功能的高级安全性
- 使用 Lambda 触发器自定义用户池工作流
- 使用 Amazon Pinpoint 进行用户池分析

有关 Amazon Cognito 身份验证和授权模型的概览，请参阅身份验证如何与 Amazon Cognito 配合使用。

要在成功进行用户池身份验证 Amazon Web Services 服务 后访问其他，请参阅在登录后使用身份池访问 Amazon Web Services 服务。

除了使用 Amazon Web Services Management Console 和用户池外 SDKs，您还可以使用来管理您的用户池Amazon Command Line Interface。

主题

- 向用户池添加社交登录
- 添加 SAML 2.0 身份提供者

向用户池添加社交登录

让用户能够通过其现有的公共或社交身份提供者登录您的应用程序，这样可以改善他们的身份验证体验。Amazon Cognito 用户池与 Facebook、Google、Amazon 和 Apple 等热门社交身份提供商 (IdPs) 集成，为您的用户提供了他们已经熟悉的便捷登录选项。

设置社交登录时，您为用户提供了一种备选方案，为您的应用程序创建一个专用账户。这样可以提高转化率，使注册过程更加顺畅。从用户的角度来看，他们可以应用现有的社交凭证来快速进行身份验证，而无需记住另一个用户名和密码。

在用户池中配置社交 IdP 涉及几个关键步骤。您必须向社交服务提供者注册您的应用程序才能获得客户端 ID 和密钥。然后，您可以将社交 IdP 配置添加到用户池中，指定要请求的范围以及要从 IdP 属性映射的用户池属性。在运行时，Amazon Cognito 会处理与提供者的令牌交换、映射用户属性，并以共享用户池格式向应用程序发放令牌。

向社交 IdP 注册

在使用 Amazon Cognito 创建社交 IdP 之前，必须向社交 IdP 注册应用程序才能接收客户端 ID 和客户端密钥。

向 Facebook 注册应用程序

1. 创建 Facebook 开发人员账户。
2. 使用 Facebook 凭证登录。
3. 在 My Apps（我的应用程序）菜单上，选择 Create New App（创建新的应用程序）。

   如果您没有现有 Facebook 应用程序，则会看到另一个选项。选择 Create App (创建应用程序)。
4. 在创建应用程序页面上，为您的应用程序选择一个用例，然后选择下一步。
5. 输入 Facebook 应用程序的名称，然后选择创建应用程序。
6. 在左侧导航栏上，选择应用程序设置，然后选择基本。
7. 记下 App ID（应用程序 ID）和 App Secret（应用程序密钥）。您将在下一节中使用它们。
8. 从页面底部选择 + 添加平台。
9. 在选择平台屏幕上，选择您的平台，然后选择下一步。
10. 选择保存更改。
11. 为 App Domains（应用程序域）输入用户池域。

    ```
    https://your_user_pool_domain
    ```
12. 选择保存更改。
13. 从导航栏中，选择产品，然后从 Facebook 登录中选择配置。
14. 从 Facebook 登录的配置菜单中，选择设置。

    在 “有效重定向” 中输入您的 OAuth 重定向 URL URIs。重定向 URL 包含具有 `/oauth2/idpresponse` 端点的用户池域。

    ```
    https://your_user_pool_domain/oauth2/idpresponse
    ```
15. 选择保存更改。

向 Amazon 注册应用程序

1. 创建 Amazon 开发人员账户。

2. 使用 Amazon 凭证登录。
3. 您需要创建一个 Amazon 安全配置文件才能接收 Amazon 客户端 ID 和客户端密钥。

   从页面顶部的导航栏中选择应用程序和服务，然后选择 Login with Amazon。
4. 选择 Create a Security Profile（创建安全配置文件）。
5. 输入 Security Profile Name（安全配置文件名称）、Security Profile Description（安全配置文件描述）和 Consent Privacy Notice URL（同意隐私声明 URL）。
6. 选择 Save（保存）。
7. 选择 Client ID（客户端 ID）和 Client Secret（客户端密钥）以显示客户端 ID 和密钥。您将在下一节中使用它们。
8. 将鼠标悬停在齿轮图标上并选择 Web Settings（Web 设置），然后选择 Edit（编辑）。
9. 将用户池域输入到 Allowed Origins（允许的源）中。

```
https://<your-user-pool-domain>
```

10. 在 “允许的返回” 中输入您的用户池域和`/oauth2/idpresponse`终端节点 URLs。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

11. 选择保存。

向 Google 注册应用程序

有关 Google Cloud 平台中 OAuth 2.0 的更多信息，请参阅 Google Workspace 开发者版文档中的了解身份验证和授权。

1. 创建 Google 开发人员账户。
2. 登录到 Google Cloud Platform 控制台。
3. 从顶部导航栏中，选择 Select a project（选择项目）。如果您在 Google 平台中已经有项目，则此菜单会改为显示您的默认项目。
4. 选择 NEW PROJECT（新建项目）。
5. 输入产品的名称，然后选择 CREATE（创建）。
6. 在左侧导航栏上，选择APIs 和服务，然后选择 Oauth同意屏幕。
7. 输入应用程序信息，包括应用程序域、授权域和开发人员联系信息。授权域必须包括 `amazoncognito.com` 和自定义域的根。例如：`example.com`。选择 SAVE AND CONTINUE（保存并继续）。

8. 1. 在 “范围” 下，选择 “添加或移除范围”，然后至少选择以下 OAuth范围。

   1. `.../auth/userinfo.email`
   2. `.../auth/userinfo.profile`
   3. openid

9. 在 Test users（测试用户）下，选择 Add users（添加用户）。输入您的电子邮件地址和任何其他授权测试用户，然后选择保存并继续。
10. 再次展开左侧导航栏，选择APIs 和服务，然后选择凭证。
11. 选择 “创建凭据”，然后选择 “OAuth 客户端 ID”。
12. 选择 Application type（应用程序类型）并为客户端提供 Name（名称）。
13. 在 “授权 JavaScript 来源” 下，选择 “添加 URI”。输入用户群体域。

```
https://<your-user-pool-domain>
```

14. 在 “授权重定向” 下 URIs，选择 “添加 URI”。输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

```
https://<your-user-pool-domain>/oauth2/idpresponse
```

15. 选择 “创建”。
16. 安全地存储 Google 在您的客户端 ID 和您的客户端密钥下显示的值。当您添加 Google IdP 时，请向 Amazon Cognito 提供这些值。

向 Apple 注册应用程序

有关设置“通过 Apple 登录”的更多信息，请参阅 Apple 开发人员文档中的配置环境以通过 Apple 登录。

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。
3. 在左侧导航栏上，选择 Certificates, Identifiers & Profiles（证书、标识符和配置文件）。
4. 在左侧导航栏上，选择 Identifiers (标识符)。
5. 在 Identifiers (标识符) 页面上，选择 + 图标。
6. 在 “注册新标识符” 页面上，选择 “应用程序” IDs，然后选择 “继续”。
7. 在选择类型页面上，选择应用程序，然后选择继续。
8. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：

1. 在 Description（说明）下，输入说明。
2. 在 App ID Prefix（应用程序 ID 前缀）下，输入 Bundle ID（捆绑包 ID）。记下 App ID Prefix（应用程序 ID 前缀）下的值。在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供商后，您将使用此值。
3. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。
4. 在 “使用 Apple 登录：应用程序 ID 配置” 页面上，选择将应用程序设置为主应用程序或与其他应用程序分组 IDs，然后选择 “保存”。
5. 选择 Continue（继续）。

9. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。
10. 在 Identifiers (标识符) 页面上，选择 + 图标。
11. 在 “注册新标识符” 页上，选择 “服务” IDs，然后选择 “继续”。
12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：

    1. 在 Description（说明）下，输入说明。
    2. 在 Identifier（标识符）下，输入标识符。记下此服务 ID，因为在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供者后需要此值。
    3. 选择 Continue (继续)，然后选择 Register (注册)。

13. 从标识符页面选择刚创建的服务 ID。

    1. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。
    2. 在 Web Authentication Configuration（Web 身份验证配置）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。
    3. 选择 “网站” 旁边的 “+” 图标 URLs。
    4. 在 Domains and subdomains（域名和子域）下，输入不带 `https://` 前缀的用户群体域。

       ```
       <your-user-pool-domain>
       ```

    5. 在 Return 下 URLs，输入您的用户池域`/oauth2/idpresponse`终端节点的路径。

       ```
       https://<your-user-pool-domain>/oauth2/idpresponse
       ```

    6. 选择下一步，然后选择完成。您不需要验证域。
    7. 选择 Continue（继续），然后选择 Save（保存）。

14. 在左侧导航栏上，选择 Keys (密钥)。

15. 在 Keys（密钥）页面上，选择 + 图标。
16. 在 Register a New Key（注册新密钥）页面上，执行以下操作：
    1. 在 Key Name（密钥名称）下，输入密钥名称。
    2. 选择 Sign In with Apple，然后选择 Configure（配置）。
    3. 在配置密钥页面上，选择您先前创建的应用程序 ID 作为主应用程序 ID。选择保存。
    4. 选择 Continue（继续），然后选择 Register（注册）。
17. 在下载您的密钥页面上，选择下载来下载私有密钥，并记下显示的密钥 ID，然后选择完成。在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供商后，您将需要此私有密钥和在此页面上显示的 Key ID（密钥 ID）值。

将社交 IdP 添加到用户池

在本节中，您使用上一节中的客户端 ID 和客户端密钥在用户池中配置社交 IdP。

使用配置用户池社交身份提供商 Amazon Web Services Management Console

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “社交和外部提供商” 菜单。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择一个社交身份提供商：Facebook、Google、Login with Amazon 或 Sign in with Apple。
6. 根据您选择的社交身份提供商，从以下步骤中进行选择：
    - Google 和 Login with Amazon – 输入在上一部分中生成的应用程序客户端 ID 和应用程序客户端密钥。
    - Facebook – 输入在上一部分中生成的应用程序客户端 ID 和应用程序客户端密钥，然后选择 API 版本（例如，版本 2.12）。我们建议选择最新的可用版本，因为每个 Facebook API 都有一个生命周期和一个弃用日期。Facebook 的范围和属性可能因 API 版本而异。我们建议您使用 Facebook 测试您的社交身份登录，以确保联合会按预期运行。
    - Sign in with Apple– 输入在上一部分中生成的服务 ID、团队 ID、密钥 ID 和私有密钥。
7. 输入要使用的授权范围的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。对于 Facebook，这些属性应用逗号分隔。对于 Google 和 Login with Amazon，则应采用空格分隔。对于 Sign in with Apple，选中要访问的范围的复选框。

社交身份提供商	示例范围
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Sign in with Apple	`email name`

您的应用程序用户需要同意向您的应用程序提供这些属性。关于社交服务提供商范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign in with Apple 的文档。

对于 Sign in with Apple，下面提供了可能不会返回范围的用户场景：

- 终端用户离开 Apple 登录页面后出现故障（可能来自 Amazon Cognito 内部的故障或开发人员编写的任何内容）。
- 服务 ID 标识符可在不同用户池和/或其他身份验证服务中使用。
- 开发人员在用户登录后添加额外的范围。用户仅在进行身份验证和刷新令牌时检索新信息。
- 开发人员删除用户，然后用户再次登录，而没有从其 Apple ID 配置文件中删除该应用程序。

8. 请将身份提供商的属性映射到您的用户池。有关更多信息，请参阅 有关映射的需知信息。
9. 选择创建。
10. 从应用程序客户端菜单中，选择列表中的一个应用程序客户端，然后编辑托管用户界面设置。将新的社交身份提供商添加到 Identity providers（身份提供商）下的应用程序客户端。
11. 选择保存更改。

测试社交 IdP 配置

可以通过使用前两节中的元素来创建登录 URL。使用此 URL 测试社交 IdP 配置。

```
https://mydomain.auth.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

您可以在用户池 Domain name（域名）控制台页上找到您的域。client_id 位于 App client settings（应用程序客户端设置）页上。对于 redirect_uri 参数，使用您的回调 URL。这是页面的 URL，在页面中，您的用户在身份验证成功后将被重定向。

> **Note**
>
> Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管登录。页面随即显示 `Something went wrong` 错误消息。

添加 SAML 2.0 身份提供者

您的应用程序用户可以通过 SAML 2.0 身份提供者（IdP）进行登录。 IdPs 当您的客户是组织的内部客户或关联企业时，您可以选择 SAML 2.0 IdPs 而不是社交。如果社交 IdP 允许所有用户注册一个账户，那么 SAML IdP 更有可能与您的组织控制的用户目录配对。无论您的用户是直接登录还是通过第三方登录，所有用户都在用户池中有一个配置文件。如果您不想添加通过 SAML 身份提供商进行的登录，请跳过此步骤。

有关更多信息，请参阅 将 SAML 身份提供者与用户池配合使用。

您必须更新 SAML 身份提供者并配置用户池。有关如何将您的用户池添加为 SAML 2.0 身份提供商的依赖方或应用程序的信息，请参阅 SAML 身份提供者的文档。

还必须向 SAML 身份提供者提供断言使用者服务（ACS）端点。在用户群体域中为您的 SAML 身份提供者中的 SAML 2.0 POST 绑定配置以下端点。有关用户池域的更多信息，请参阅 配置用户池域。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://<yourDomainPrefix>.auth.<region>.amazoncognito.com/saml2/idpresponse
With a custom domain:
https://Your custom domain/saml2/idpresponse
```

您可以在 Amazon Cognito 控制台的域名菜单中找到您的域名前缀和用户池的区域值。

对于一些 SAML 身份提供者，您还需要提供服务提供者（SP）urn，也称为受众 URI 或 SP 实体 ID，采用以下格式：

```
urn:amazon:cognito:sp:<yourUserPoolID>
```

您可以在 Amazon Cognito 控制台的用户池概览控制面板中找到您的用户池 ID。

您还应配置 SAML 身份提供商，以便为您的用户池中需要的任何属性提供属性值。通常情况下，`email` 是用户池的必需属性。在这种情况下，SAML 身份提供商应在 SAML 断言中提供一个 `email` 值（声明）。

Amazon Cognito 用户池支持 SAML 2.0 与 POST 绑定端点联合。这使您的应用程序不必检索或分析 SAML 断言响应，因为用户池直接通过用户代理从身份提供者接收 SAML 响应。

在您的用户池中配置 SAML 2.0 身份提供商

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “社交和外部提供商” 菜单。找到 Federated sign-in（联合登录），然后选择 Add an identity provider（添加身份提供商）。
5. 选择 SAML 社交身份提供商。
6. 输入以逗号分隔的 Identifiers（标识符）。标识符将告知 Amazon Cognito 应该检查用户登录时输入的电子邮件地址。然后将它们引导到与其域对应的提供者。
7. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。您必须将 SAML 2.0 身份提供商配置为向配置托管登录时创建的`https://<your Amazon Cognito domain>/saml2/logout`终端节点发送注销响应。`saml2/logout` 端点使用 POST 绑定。

   > **Note**
   >
   > 如果选择此选项，并且您的 SAML 身份提供者需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
   > SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。

8. 选择 Metadata document source（元数据文档源）。如果您的身份提供商在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

> **Note**
>
> 如果您的提供者具有公有端点，我们建议您输入元数据文档 URL，而不是上传文件。这样就可让 Amazon Cognito 自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。

9. 选择 Map attributes between your SAML provider and your app（在 SAML 提供商和应用程序之间映射属性）将 SAML 提供程序属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

   例如，当您选择 User pool attribute（用户池属性）`email` 时，按照您的身份提供商提供的 SAML 断言中显示的内容，输入 SAML 属性名称。您的身份提供商可能会提供示例 SAML 断言以供参考。一些身份提供商使用简单名称（如 `email`），另一些则使用以下示例 URL 格式的属性名称：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. 选择创建。

Amazon Cognito 身份池入门

借助 Amazon Cognito 身份池，您可以创建唯一身份并为用户分配权限。身份池可以从以下类型的身份验证服务中引入身份：

- Amazon Cognito 用户池中的用户
- 通过外部身份提供商（例如 Facebook、Google、Apple 或 OIDC 或 SAML 身份提供商）进行身份验证的用户。
- 通过您自己的现有身份验证流程进行身份验证的用户

用户向其提供商进行身份验证并向身份池出示授权后，他们将获得临时 Amazon 证书。用户的凭证具有您为访问其他 Amazon Web Services 服务而定义的权限。

主题

- 在 Amazon Cognito 中创建一个身份池
- 设置 SDK
- 集成身份提供商
- 获取凭证

在 Amazon Cognito 中创建一个身份池

您可以通过 Amazon Cognito 控制台创建身份池，也可以使用（Amazon Command Line Interface CLI) 或 Amazon Cognito。 APIs以下过程是在控制台中创建新身份池的通用指南。您也可以直接跳到控制台，按照引导式体验和内联帮助内容进行操作。

在控制台中创建新的身份池

1. 登录 Amazon Cognito 控制台并选择身份池。要向您的 IAM 委托人分配权限，以便他们可以创建和管理 Amazon Cognito 资源，请参阅。Amazon 亚马逊 Cognito 的托管策略该`AmazonCognitoPowerUser`策略足以创建身份池。
2. 选择创建身份池。
3. 在配置身份池信任中，选择将您的身份池设置为经过身份验证的访问权限和/或访客访问权限。

- 如果您选择了经过身份验证的访问权限，请在身份池中选择一个或多个您要设置为经过身份验证的身份来源的身份类型。如果您配置了自定义开发人员提供者，则在创建身份池后无法对其进行修改或删除。

4. 在配置权限中，为身份池中经过身份验证的用户或访客用户选择原定设置 IAM 角色。

   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

   b. 如果您的 Amazon Web Services 账户 角色中已有要使用的角色，则可以选择使用现有 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅 角色信任和权限。

5. 在 Connect 身份提供商中，输入您在配置身份池信任中选择的身份提供商 (IdPs) 的详细信息。系统可能会要求您提供 OAuth 应用程序客户端信息、选择 Amazon Cognito 用户池、选择 IAM IdP 或输入开发者提供商的自定义标识符。

   a. 为每个 IdP 选择角色设置。您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

   b. 您可以为每个 IdP 配置访问控制属性。访问控制属性将用户声明映射到 Amazon Cognito 应用于其临时会话的主体标签。您可以构建 IAM policy，以根据应用于用户会话的标签来筛选用户访问权限。

      i. 如果不应用主体标签，请选择非活动。

      ii. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。

      iii. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

6. 在配置属性中，在身份池名称下输入名称。
7. 在基本（经典）身份验证下，选择是否要激活基本流程。启用基本流程后，您可以绕过为自己选择的角色 IdPs ，AssumeRoleWithWebIdentity直接致电。有关更多信息，请参阅 身份池身份验证流程。
8. 如果要将标签应用到身份池，请在标签下选择添加标签。
9. 在查看并创建中，确认您为新身份池所做的选择。选择编辑以返回向导并更改任何设置。完成后，选择创建身份池。

设置 SDK

要使用 Amazon Cognito 身份池，请设置 Amazon Amplify 适用于 Java 的 Amazon SDK、或。 适用于 .NET 的 Amazon SDK有关更多信息，请参阅以下主题。

- 在《适用于 JavaScript 的 Amazon SDK 开发者指南》 JavaScript中设置 SDK
- 《Amplify 开发中心》中的 Amplify 文档
- 《适用于 .NET 的 Amazon SDK 开发人员指南》中的 Amazon Cognito 凭证提供者

集成身份提供商

Amazon Cognito 身份池（联合身份）通过 Amazon Cognito 用户群体、联合身份提供者（包括 Amazon、Facebook、Google、Apple 和 SAML 身份提供者）以及未经身份验证的身份支持用户身份验证。此功能还支持 经开发人员验证的身份，这让您能够通过自己的后端身份验证流程注册并对用户进行身份验证。

要了解有关使用 Amazon Cognito 用户池创建自己的用户目录的更多信息，请参阅Amazon Cognito 用户群体和在登录后使用身份池访问 Amazon Web Services 服务。

要了解有关使用外部身份提供商的更多信息，请参阅 身份池第三方身份提供者。

要了解有关集成自己的后端身份验证流程的更多信息，请参阅经开发人员验证的身份。

获取凭证

Amazon Cognito 身份池为访客用户（未经身份验证）和已通过身份验证并收到令牌的用户提供临时 Amazon 证书。有了这些 Amazon 证书，您的应用程序就可以 Amazon 通过 Amazon API Gateway 安全地访问内部 Amazon 或外部的后端。请参阅获取凭证。

Amazon Cognito 的引导式设置选项

您可能希望在结构化的指导性界面中评估 Amazon Cognito 的特征。以下是一些外部资源，可提供针对用户池和身份池的定制化体验。

完成讲习会

Amazon 研讨会工作室举办研讨会，引导你完成大部分 Amazon Cognito 功能的设置。这些特征包括用户池 API、用户池托管 UI、身份池和安全配置。

添加示例中的应用程序代码

本指南中的代码示例章节包含可用于用户池和身份池的应用程序代码。代码示例章节的用户池部分包含涵盖各个操作的简短片段和较长的示例， end-to-end例如使用各种编程语言的应用程序。

使用创建全栈应用程序 Amazon Amplify

Amazon Amplify Amazon Web Services 服务 适用于想要开发和托管应用程序和用户界面的开发人员。Amazon Cognito 是 Amplify 的身份验证组件。当您在应用程序中添加身份验证时，Amplify 可以自动部署 Amazon Cognito 用户池和身份池资源。另请参阅将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成。

更多亚马逊 Cognito 应用程序资源请访问 GitHub

- 适用于 Amazon Cognito 的 .NET 身份验证流示例
- Amazon Cognito 无密码身份验证
- PetStore带有 Amazon 验证权限的示例
- 使用 ABAC + 身份池访问 Amazon 资源的 React 应用程序示例
- 使用 CDK 基于 Amazon Cognito 和 API Gateway 的机器到机器授权 Amazon
- 使用 Amazon Cognito、API Gateway 和 IAM 构建精细授权
- CloudFront授权 @edge

更多讲习会

- 使用 Amazon Cognito 实现无密码身份验证和 WebAuthn
- 亚马逊 Cognito 研讨会
- 亚马逊 Cognito 故障排除研讨会

- 使用 Amazon Cognito 进行身份验证和授权以及经过验证的权限
- Amazon Cognito JWT 深入探究

博客文章

- 使用亚马逊代理保护亚马逊 Cognito 的公共客户端 CloudFront
- 如何设置 Amazon Cognito 以使用 Azure AD 进行联合身份验证
- 简化 Web 应用程序身份验证：使用 Amazon Cognito 用户池进行 AD FS 联合身份验证的指南

将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成

Amazon Cognito 的实现是在应用程序中混合使用 Amazon Web Services Management Console 我们的 Amazon 软件开发工具包管理工具和软件开发工具包库。Amazon Cognito 控制台是用于设置和管理 Amazon Cognito 用户群体和身份池的可视界面。

您可以使用 Amazon Cognito 用户池创建最省力的集成是托管登录。托管登录是一款 ready-to-use基于 Web 的登录应用程序，用于快速测试和部署 Amazon Cognito 用户池。使用托管登录进行用户池身份验证需要使用引导用户访问托管登录页面的 OpenID Connect (OIDC) 库。在这一系列用户交互式和重定向网络终端节点中，Amazon Cognito 负责处理身份验证流程，包括第三方登录、多因素身份验证 (MFA) 和选择身份验证流程。您的应用程序只需要处理 Amazon Cognito 在响应中返回的身份验证结果即可。

您还可以向应用程序添加 S Amazon DK、自定义构建身份验证接口，并调用 API 操作对用户进行身份验证和授权。 Amazon Amplify Amazon Web Services 服务 用于构建全栈应用程序，后端使用 Amazon Cognito 身份验证。

例如，您的应用程序可能会调用托管登录进行用户登录，然后从您的应用程序代码中调用令牌端点，将用户的授权码换成令牌。然后，应用程序必须解释和存储用户的令牌，并在适当的上下文中出示它们以进行身份验证和授权。Amplify 为这些流程添加了带有内置功能的引导式集成工具。

您也可以完全在代码中构建您的 Amazon Cognito 资源。身份池的托管身份验证选项与用户池不同，要访问应用程序中的 Amazon 证书，请在导入的 SDK 模块中实现身份池操作。要开始使用自己的自定义应用程序代码，请访问 Amazon Cognito 代码示例。Amazon SDKs要将 Amazon Cognito 作为 OpenID Connect 身份提供商进行集成，请使用 OpenID Connect 开发工具。

在使用 Amazon Cognito 进行身份验证和授权之前，请选择应用程序平台并准备代码以与服务集成。有关可用的平台 Amazon SDKs，请参阅使用进行身份验证 Amazon SDKs。 Amazon CLI 是适用于 Amazon Cognito Amazon Web Services 服务等的命令行软件开发工具包，是开始熟悉 Amazon Cognito API 操作及其语法的宝贵场所。

> **Note**
>
> Amazon Cognito 的某些组件只能使用 API 进行配置。例如，您只能使用更新或 UpdateUserPoolAPI 请求中UserPool类`LambdaConfig`属性的请求来设置用户池自定义 SMS 或电子邮件发件人 Lambda 触发器。CreateUserPool

Amazon Cognito 用户群体 API 与多个 API 操作类共享其命名空间。一个类配置用户群体及其进程、身份提供商和用户。另一个类包括用户在公共客户端中执行的未经身份验证的操作，旨在登录、注销和管理他们的配置文件。最后一类 API 操作在机密的服务器端客户端中执行您使用自己的 Amazon 凭据授权的用户操作。在开始实现应用程序代码之前，您必须知道您想要的应用程序架构。有关更多信息，请参阅 了解 API、OIDC 和托管登录页面身份验证。

主题

- 使用进行身份验证 Amazon Amplify
- 使用进行身份验证 Amazon SDKs
- 身份验证如何与 Amazon Cognito 配合使用
- 将此服务与 Amazon SDK 配合使用
- 使用 Amazon Verified Permissions 进行授权

使用进行身份验证 Amazon Amplify

Amazon Amplify 是用于构建 Web 和移动应用程序的完整解决方案。借助 Amplify，您可以使用 Amplify 库连接到现有资源，也可以使用 Amplify 命令行界面（CLI）创建和配置新资源。Amplify 还连接了用户界面组件（如身份验证器），以便在您的应用程序中设置和自定义登录和注册体验。

要在前端应用程序中使用 Amplify 身份验证功能，请参阅以下按平台分列的文档。

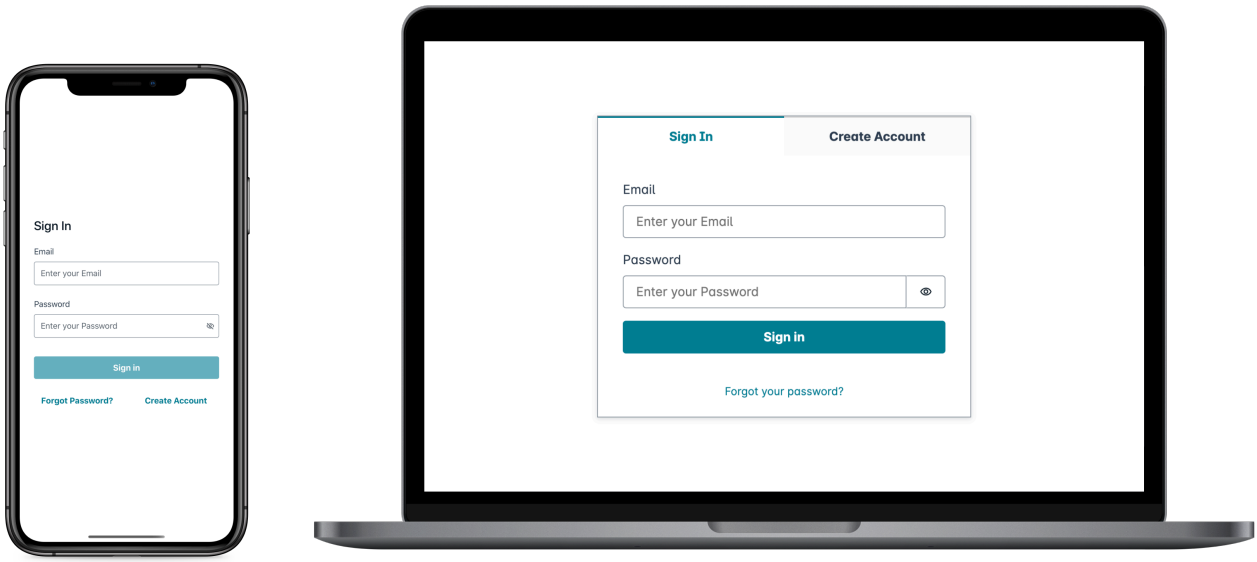
- React 的 Amplify 身份验证
- React Native 的 Amplify 身份验证
- 适用于 Swift 的 Amplify 身份验证 (iOS)
- 适用于 Android 的 Amplify 身份验证
- 放大 Flutter 的身份验证

Amplify 库是开源的，可在上使用。GitHub要了解有关 Amplify Auth 如何实现 Amazon Cognito 身份验证的更多信息，请访问以下库。

- amplify-js
- amplify-swift
- amplify-flutter
- amplify-android

使用 Amplify 创建用户界面（UI）

用户池托管登录可以满足 Web 或移动应用程序身份验证前端的基本需求。要在托管登录所能容纳的参数之外自定义用户界面 (UI)，请自定义构建应用程序。Amplify UI 是各种语言的前端组件的可自定义集合。



要开始使用您的自定义身份验证组件，请访问身份验证器组件的以下文档。

- 适用于 Android 的身份验证器
- 适用于 Angular 的身份验证器
- 适用于 Flutter 的身份验证器
- 适用于 React 的身份验证器
- 适用于 React Native 的身份验证器
- 适用于 Swift 的身份验证器
- 适用于 Vue 的身份验证器

使用进行身份验证 Amazon SDKs

要使用安全的后端构建您自己的身份微服务以与 Amazon Cognito 交互，请使用您选择的语言版本的软件开发工具包连接到 Amazon Cognito 用户池和 Amazon Cognito 身份池 API。 Amazon

有关各个 API 操作的详细信息，请参阅 Amazon Cognito 用户池 API 参考和 Amazon Cognito API 参考。这些文档包含 “另请参阅” 章节，其中包含 SDKs在支持的平台中使用各种资源的资源。

- Amazon 命令行界面
- Amazon 适用于 .NET 的 SDK
- Amazon 适用于 C++ 的 SDK
- Amazon 适用于 Go 的 SDK
- Amazon 适用于 Java 的 SDK V2
- Amazon 适用于 JavaScript
- Amazon 适用于 PHP 的 SDK V3
- Amazon Python 软件开发工具包
- Amazon 适用于 Ruby V3 的 SDK

身份验证如何与 Amazon Cognito 配合使用

当您的客户登录 Amazon Cognito 用户池时，您的应用程序会收到 JSON 网络令牌 () JWTs。

当您的客户使用用户池令牌或其他提供商登录身份池时，您的应用程序将收到临时 Amazon 证书。

通过用户池登录，您可以完全使用 Amazon SDK 实现身份验证和授权。如果您不想构建自己的用户界面 (UI) 组件，则可以调用预构建的 Web UI（托管登录）或第三方身份提供商 (IdP) 的登录页面。

本主题概述了您的应用程序与 Amazon Cognito 交互以使用 ID 令牌进行身份验证、使用访问令牌进行授权以及使用身份池 Amazon Web Services 服务 凭证进行访问的一些方式。

主题

- 使用托管登录进行用户池身份验证
- 使用 Amazon SDK 进行用户池 API 身份验证和授权
- 使用第三方身份提供者进行用户池身份验证
- 身份池身份验证

使用托管登录进行用户池身份验证

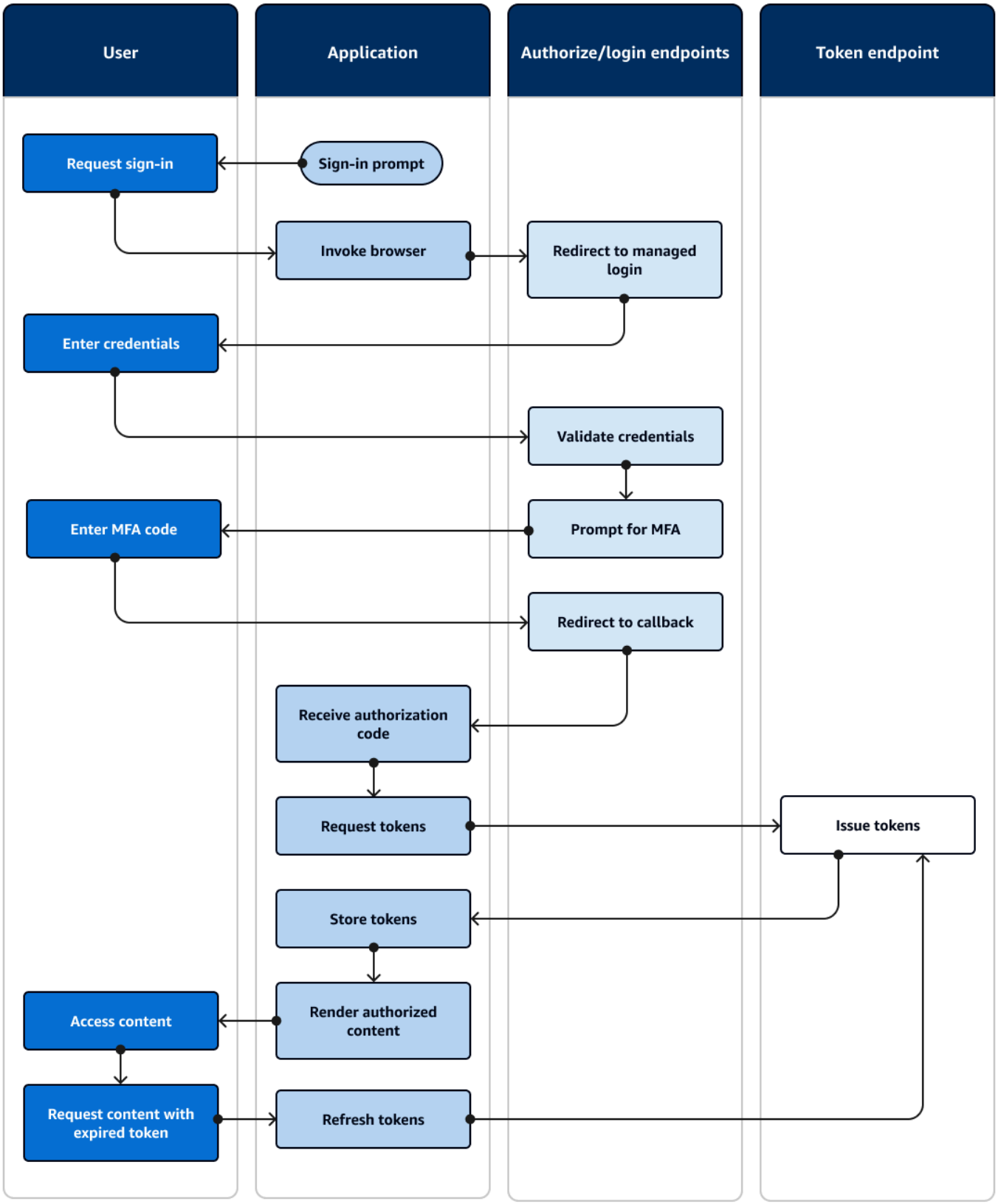
托管登录是一个链接到您的用户池和应用程序客户端的网站。使用它可为用户执行登录、注册和密码重置操作。具有用于身份验证的托管登录组件的应用程序可能需要较少的开发人员来实现。应用程序可以跳过用户界面组件进行身份验证，并在用户的浏览器中调用托管登录网页。

应用程序 JWTs 通过网络或应用程序重定向位置收集用户。实现托管登录的应用程序可以连接到用户池进行身份验证，就像它们是 OpenID Connect (OIDC) IdP 一样。

托管登录符合这样的模式，即应用程序需要 OIDC 授权服务器的身份验证服务，但不立即需要自定义身份验证、身份池集成或用户属性自助服务等功能。当您想使用其中一些高级选项时，您可以使用 SDK 的用户池组件来实施。

托管登录和第三方 IdP 身份验证模型主要依赖 OIDC 的实现，最适合范围为 2.0 的高级授权模型。OAuth

下图说明了托管登录身份验证的典型登录会话。


User
Application
Authorize/login endpoints
Token endpoint
Request sign-in
Sign-in prompt
Invoke browser
Redirect to managed login
Enter credentials
Validate credentials
Enter MFA code
Prompt for MFA
Redirect to callback
Receive authorization code
Request tokens
Issue tokens
Store tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

托管登录身份验证流程

1. 一个用户访问您的应用程序。
2. 他们选择“登录”链接。
3. 该应用程序会将用户引导至用户池域的托管登录页面中的登录提示。
4. 他们输入用户名和密码。
5. 用户池验证用户的凭证，并确定用户具有已激活的多重身份验证（MFA）。
6. 托管登录页面会提示用户输入 MFA 代码。
7. 用户输入他们的 MFA 代码。
8. 您的用户池将用户重定向到应用程序 URL。
9. 应用程序从附加到回调 URL 的托管登录的 URL 请求参数中收集授权码。
10. 应用程序通过授权代码请求令牌。
11. 令牌端点返回 JWTs 到应用程序。
12. 应用程序解码、验证、存储或缓存用户的。 JWTs
13. 应用程序显示请求的访问控制组件。
14. 用户查看其内容。
15. 后来，用户的访问令牌过期，他们请求查看访问控制组件。
16. 应用程序确定用户的会话应该持续下去。应用程序使用刷新令牌从令牌端点请求新令牌。

变体和自定义

您可以使用适用于整个用户群的品牌编辑器自定义托管登录页面的外观，也可以在任何应用程序客户端级别上自定义托管登录页面的外观。您还可以使用自己的身份提供者、范围、用户属性的访问权限和高级安全配置来配置应用程序客户端。

相关资源

- 用户池托管登录
- 作用域、M2M 以及 APIs 带资源服务器的场景
- 用户池端点和托管登录参考

使用 Amazon SDK 进行用户池 API 身份验证和授权

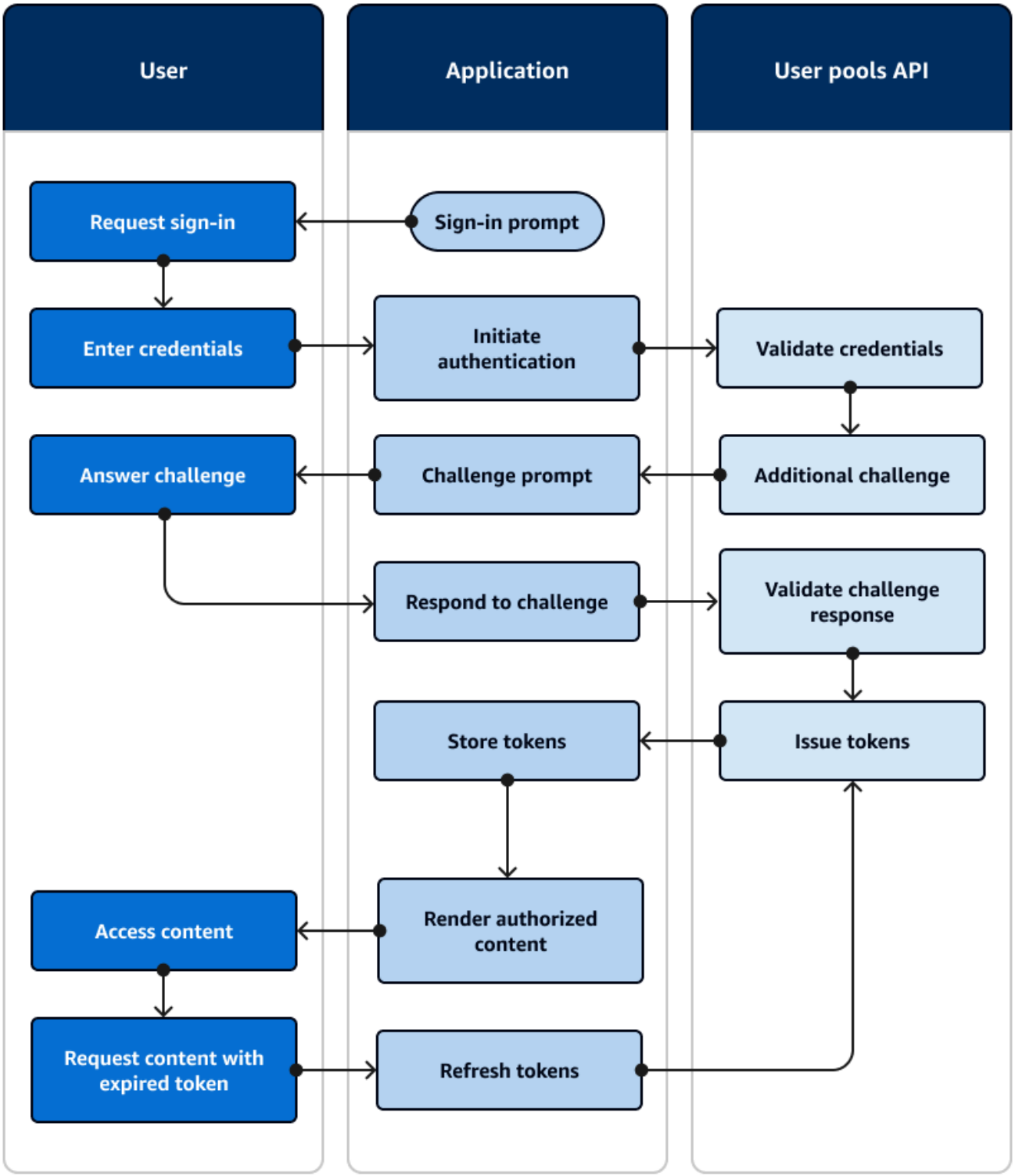
Amazon 已在各种开发者框架中为 Amazon Cognito 用户池或 Amazon Cognito 身份提供商开发了组件。其中内置的方法 SDKs 调用 Amazon Cognito 用户池 API。同一个用户池 API 命名空间同时具有用于用户池配置和用户身份验证配置的操作。有关更全面的概述，请参阅了解 API、OIDC 和托管登录页面身份验证。

API 身份验证适合您的应用程序具有现有 UI 组件且主要依赖用户池作为用户目录的模型。这种设计将 Amazon Cognito 添加为大型应用程序中的一个组件。该设计需要使用编程逻辑来处理复杂的质询和响应链。

此应用程序不需要实现完整的 OpenID Connect（OIDC）依赖方实施。相反，它具有解码和使用的 JWTs能力。如果您想访问本地用户的全套用户池特征，请在您的开发环境中使用 Amazon Cognito SDK 构建身份验证。

使用自定义 OAuth 范围的 API 身份验证不太倾向于外部 API 授权。要通过 API 身份验证向访问令牌添加自定义范围，请在运行时使用 令牌生成前 Lambda 触发器 修改令牌。

以下示意图说明了 API 身份验证的典型登录会话。


User
Application
User pools API
Request sign-in
Sign-in prompt
Enter credentials
Initiate authentication
Validate credentials
Answer challenge
Challenge prompt
Additional challenge
Respond to challenge
Validate challenge response
Store tokens
Issue tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

API 身份验证流程

1. 一个用户访问您的应用程序。
2. 他们选择“登录”链接。
3. 他们输入用户名和密码。
4. 应用程序调用发出 InitiateAuthAPI 请求的方法。该请求会将用户的凭证传递到用户池。
5. 用户池验证用户的凭证，并确定用户具有已激活的多重身份验证（MFA）。
6. 用户池通过请求获取 MFA 代码的质询进行响应。
7. 应用程序会生成一个提示，指明从用户那里收集 MFA 代码。
8. 应用程序调用发出 RespondToAuthChallengeAPI 请求的方法。请求传递用户的 MFA 代码。
9. 用户池验证用户的 MFA 代码。
10. 用户池以用户池的回应 JWTs。
11. 应用程序解码、验证、存储或缓存用户的。 JWTs
12. 应用程序显示请求的访问控制组件。
13. 用户查看其内容。
14. 后来，用户的访问令牌过期，他们请求查看访问控制组件。
15. 应用程序确定用户的会话应该持续下去。它使用刷新令牌再次调用该InitiateAuth方法并检索新令牌。

变体和自定义

您可以通过额外质询（例如，您自己的自定义身份验证质询）来增强此流程。您可以自动限制以下用户的访问权限：其密码已泄露的用户，或者表现出意料之外的特性，可能表明存在恶意登录尝试的用户。注册、更新用户属性和重置密码的操作流程大致相同。这些流程中的大多数都有重复的公共（客户端）和机密（服务器端）API 操作。

相关资源

- Amazon Cognito 用户池 API
- 用户池入门
- 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成
- 了解 API、OIDC 和托管登录页面身份验证

使用第三方身份提供者进行用户池身份验证

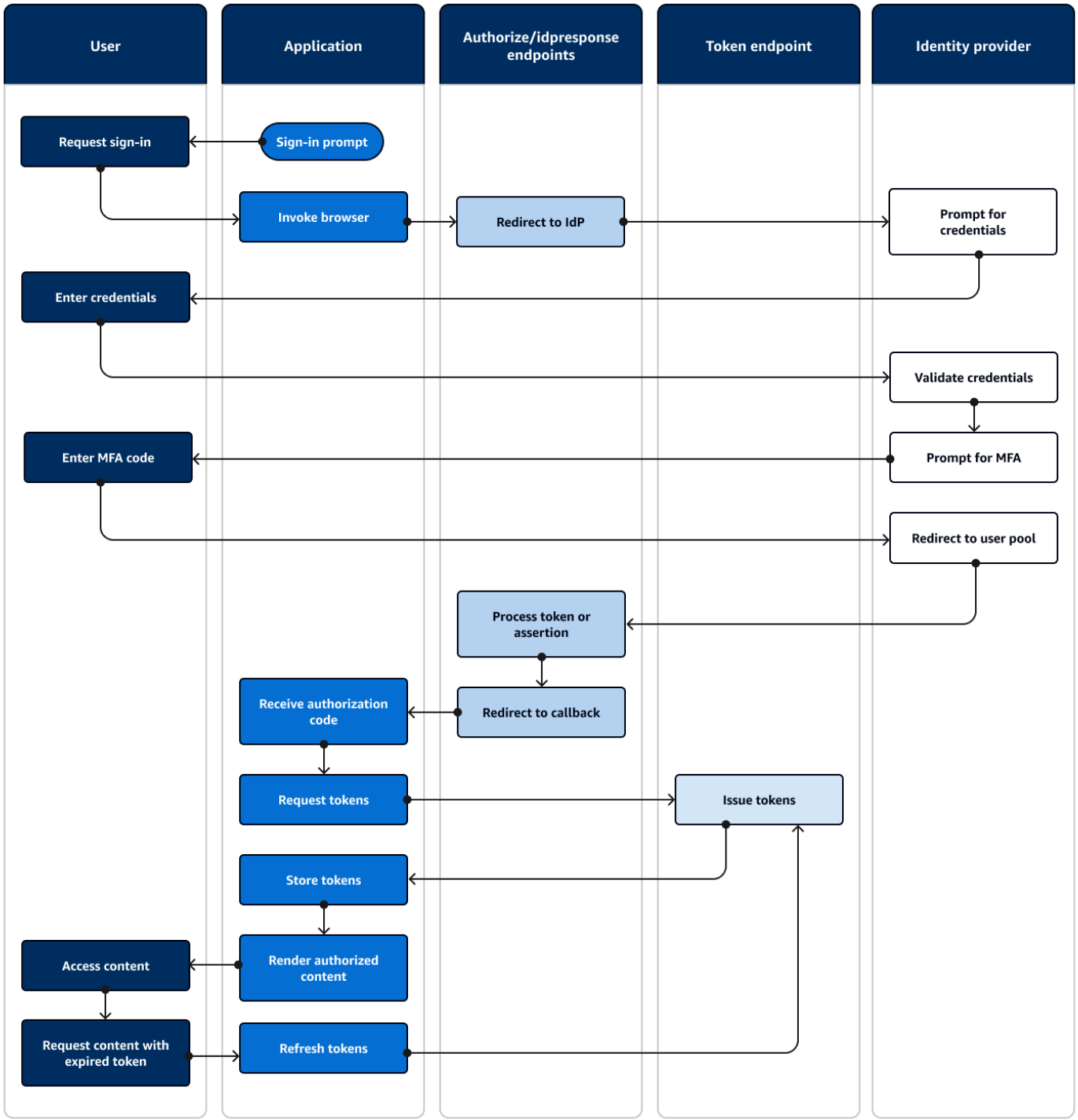
使用外部身份提供商 (IdP) 登录或联合身份验证与托管登录的模式类似。您的应用程序是用户池的 OIDC 依赖方，而您的用户池则充当 IdP 的传递方。IdP 可以是像 Facebook 或 Google 这样的使用者用户目录，也可以是像 Azure 这样的 SAML 2.0 或 OIDC 企业目录。

您的应用程序不是在用户的浏览器中进行托管登录，而是调用用户池授权服务器上的重定向端点。从用户的角度来看，他们是选择您的应用程序中的登录按钮。然后，他们的 IdP 提示他们登录。与托管登录身份验证一样，应用程序 JWTs 在应用程序的重定向位置收集。

使用第三方 IdP 进行身份验证适合这样的模型：用户在注册您的应用程序时可能不想使用新密码。可以毫不费力地将第三方身份验证添加到已实现托管登录身份验证的应用程序中。实际上，由于您在用户浏览器中调用的内容略有不同，托管登录和第三方会 IdPs 产生一致的身份验证结果。

与托管登录身份验证一样，联合身份验证最适合范围为 OAuth 2.0 的高级授权模型。

以下示意图说明了联合身份验证的典型登录会话。



联合身份验证流程

1. 一个用户访问您的应用程序。
2. 他们选择“登录”链接。

3. 应用程序将用户引导至其 IdP 的登录提示处。
4. 他们输入用户名和密码。
5. IdP 验证用户的凭证，并确定用户具有已激活的多重身份验证（MFA）。
6. IdP 会提示用户输入 MFA 代码。
7. 用户输入他们的 MFA 代码。
8. IdP 使用 SAML 响应或授权代码将用户重定向至用户池。
9. 如果用户传递了授权码，则用户池会静默地将该代码交换为 IdP 令牌。用户池会验证 IdP 令牌，并使用新的授权代码将用户重定向到应用程序。
10. 应用程序从用户池附加到回调 URL 的 URL 请求参数收集授权代码。
11. 应用程序通过授权代码请求令牌。
12. 令牌端点返回 JWTs 到应用程序。
13. 应用程序解码、验证、存储或缓存用户的。 JWTs
14. 应用程序显示请求的访问控制组件。
15. 用户查看其内容。
16. 后来，用户的访问令牌过期，他们请求查看访问控制组件。
17. 应用程序确定用户的会话应该持续下去。应用程序使用刷新令牌从令牌端点请求新令牌。

变体和自定义

您可以在托管登录中启动联合身份验证，用户可以从分配给应用程序客户端 IdPs 的列表中进行选择。托管登录还可以提示输入电子邮件地址，并自动将用户的请求路由到相应的 SAML IdP。使用第三方身份提供商进行身份验证不需要用户通过托管登录进行交互。您的应用程序可以向用户的授权服务器请求添加请求参数，并让用户静默地重定向到其 IdP 登录页面。

相关资源

- 用户池使用第三方身份提供商登录
- 作用域、M2M 以及 APIs 带资源服务器的场景
- 用户池端点和托管登录参考

身份池身份验证

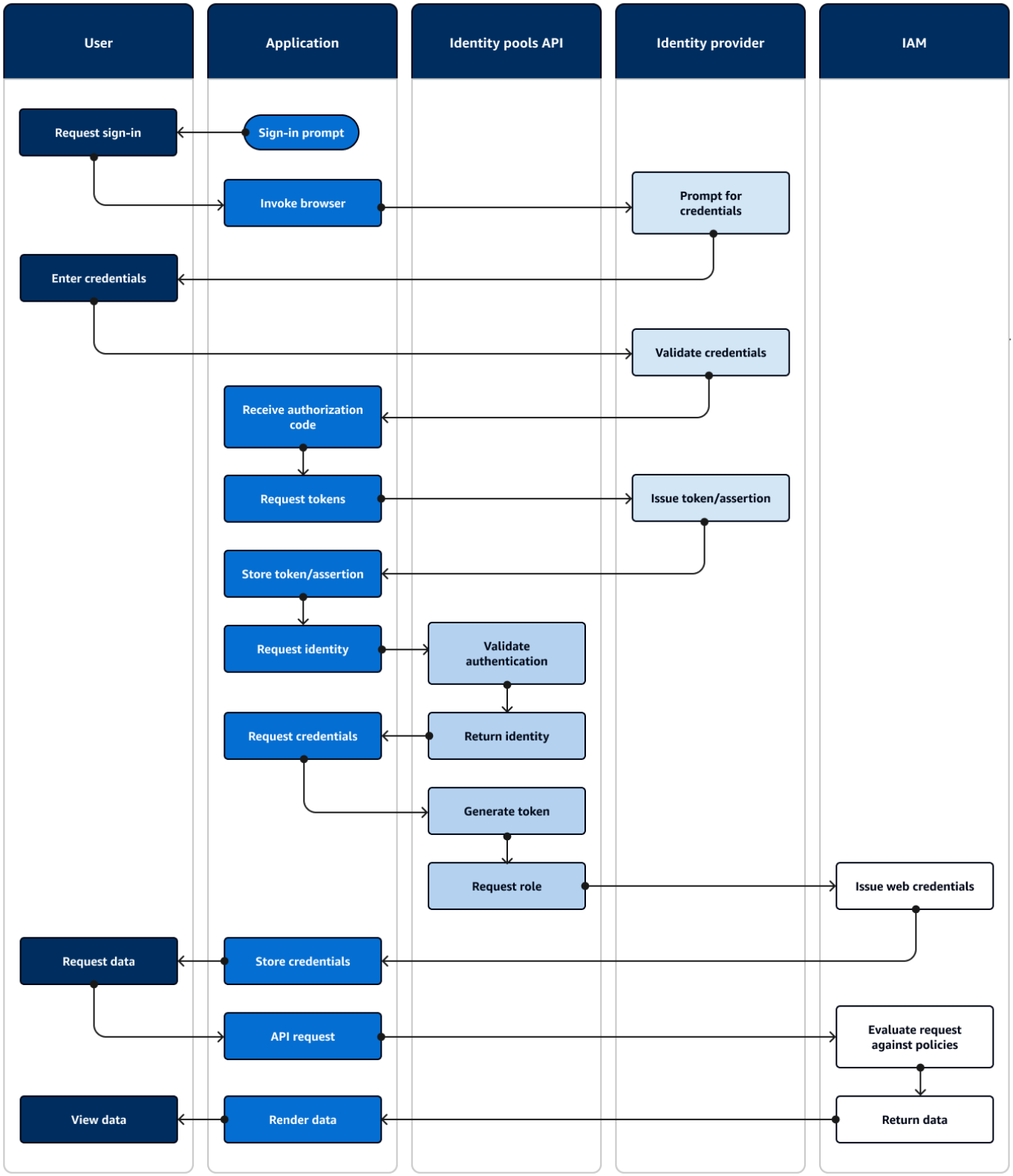
身份池是您应用程序中的一个组件，其功能、API 命名空间和 SDK 模型与用户池有所不同。用户池提供基于令牌的身份验证和授权，而身份池则为 Amazon Identity and Access Management (IAM) 提供授权。

您可以将一组分配 IdPs 给身份池，并使用这些身份池登录用户。用户池作为身份池紧密集成 IdPs ，为身份池提供了最多的访问控制选项。同时，身份池有多种身份验证选项可供选择。用户池加入 SAML、OIDC、社交、开发者和访客身份源，作为从身份池中获得临时 Amazon 证书的路由。

身份池的身份验证是外部的，它遵循前面说明的其中一个用户池流程，或者您与另一个 IdP 独立开发的流程。在您的应用程序执行初始身份验证后，它会将证明传递给身份池并收到一个临时会话作为返回。

使用身份池进行身份验证适合您使用 IAM 授权对应用程序资产和数据实施访问控制 Amazon Web Services 服务 的模式。与用户池中的 API 身份验证一样，成功的应用程序包括 Amazon SDKs 您想要访问的每项服务，以使用户受益。 Amazon SDKs 将身份池身份验证中的凭据作为签名应用于 API 请求。

以下示意图说明了使用 IdP 的身份池身份验证的典型登录会话。


User
Application
Identity pools API
Identity provider
IAM
Request sign-in
Sign-in prompt
Invoke browser
Prompt for credentials
Enter credentials
Validate credentials
Receive authorization code
Request tokens
Issue token/assertion
Store token/assertion
Request identity
Validate authentication
Request credentials
Return identity
Generate token
Request role
Issue web credentials
Request data
Store credentials
API request
Evaluate request against policies
View data
Render data
Return data

身份池身份验证流程

1. 一个用户访问您的应用程序。
2. 他们选择“登录”链接。
3. 应用程序将用户引导至其 IdP 的登录提示处。
4. 他们输入用户名和密码。
5. IdP 验证用户的凭证。
6. IdP 将用户重定向至具有 SAML 响应或授权代码的应用程序。
7. 如果用户传递了授权代码，则应用程序会将该代码交换为 IdP 令牌。
8. 应用程序解码、验证、存储或缓存用户或断言。 JWTs
9. 应用程序调用发出 GetIdAPI 请求的方法。应用程序传递用户的令牌或断言并请求身份 ID。
10. 身份池根据配置的身份提供者验证令牌或断言。
11. 身份池返回身份 ID。
12. 应用程序调用发出 GetCredentialsForIdentityAPI 请求的方法。应用程序传递用户的令牌或断言并请求 IAM 角色。
13. 身份池会生成一个新的 JWT。新的 JWT 包含请求 IAM 角色的声明。身份池根据用户的请求和 IdP 的身份池配置中的角色选择标准来确定角色。
14. Amazon Security Token Service (Amazon STS) 响应来自身份池的AssumeRoleWithWebIdentity请求。响应包含使用 IAM 角色进行临时会话的 API 凭证。
15. 应用程序存储会话凭证。
16. 用户在需要 Amazon中的访问受保护资源的应用程序中执行操作。
17. 应用程序将临时证书作为签名应用于所需的 API 请求 Amazon Web Services 服务。
18. IAM 评估凭证中附加到该角色的策略。并将策略与请求进行比较。
19. Amazon Web Services 服务 返回请求的数据。
20. 应用程序在用户的界面中呈现数据。
21. 用户查看数据。

变体和自定义

要使用用户池来可视化身份验证，请在发放令牌/断言步骤之后插入其中一个之前的用户池概述。开发人员身份验证将请求身份之前的所有步骤替换为由开发人员凭证签名的请求。访客身份验证还会直接跳到请求身份步骤，不验证身份验证，并返回 limited-access IAM 角色的凭证。

相关资源

- Amazon Cognito 身份池
- 用户 IAM 角色
- 身份池身份验证流程

将此服务与 Amazon SDK 配合使用

Amazon 软件开发套件 (SDKs) 可用于许多流行的编程语言。每个软件开发工具包都提供 API、代码示例和文档，使开发人员能够更轻松地以其首选语言构建应用程序。

SDK 文档
Amazon CLI
适用于 Java 的 Amazon SDK
适用于 JavaScript 的 Amazon SDK
适用于 .NET 的 Amazon SDK
适用于 PHP 的 Amazon SDK
Amazon Tools for PowerShell
适用于 Python (Boto3) 的 Amazon SDK
适用于 Ruby 的 Amazon SDK
适用于 SAP ABAP 的 Amazon SDK

使用 Amazon Verified Permissions 进行授权

Amazon Verified Permissions 是针对您构建的应用程序的授权服务。当您将 Amazon Cognito 用户群体添加为身份源时，应用程序可以将用户群体访问权限或身份（ID）令牌传递给 Verified Permissions，以便做出允许或拒绝决定。Verified Permissions 根据您使用 Cedar 策略语言编写的策略，来考虑用户的属性和请求上下文。请求上下文可以包括所请求的文档、映像或其他资源的标识符，以及用户想要对该资源采取的操作。

您的应用程序可以在或 BatchIsAuthorizedWithTokenAPI 请求中向已验证的权限提供用户的身份IsAuthorizedWithToken或访问令牌。这些 API 操作接受您的用户作为 `Principal` 并对他们想要访问的 `Resource` 上的 `Action` 作出授权决策。使用额外的自定义 `Context` 有助于作出细致的访问决策。

当应用程序在 `IsAuthorizedWithToken` API 请求中提供令牌时，Verified Permissions 将执行以下验证。

1. 您的用户群体是针对所请求的策略存储而配置的 Verified Permissions 身份来源。
2. 访问令牌或身份令牌中的 `client_id` 或 `aud` 声明分别与您提供给 Verified Permissions 的用户群体应用程序客户端 ID 相匹配。要验证此声明，您必须在 Verified Permissions 身份来源中配置客户端 ID 验证。
3. 您的令牌未过期。
4. 您的令牌中 `token_use` 声明的值与您传递给 `IsAuthorizedWithToken` 的参数相匹配。如果您将 `token_use` 声明传递给 `accessToken` 参数，则该声明必须是 `access`，并且如果将它传递给 `identityToken` 参数，则声明必须是 `id`。
5. 令牌中的签名来自用户池中已发布的 JSON 网络密钥 (JWKs)。你可以在 JWKs at 中查看你的`https://cognito-idp.`*`Region`*`.amazonaws.com/`*`your user pool ID`*`/.well-known/jwks.json`。

已撤销的令牌和已删除的用户

Verified Permissions 仅验证它从您的身份来源和用户令牌到期时间所了解的信息。Verified Permissions 不检查令牌是否撤销或用户是否存在。如果您从用户群体中撤销了用户的令牌或删除了用户的配置文件，则 Verified Permissions 在令牌到期之前仍会认为该令牌有效。

策略评估

将您的用户群体配置为策略存储的身份来源。将您的应用程序配置为在对 Verified Permissions 的请求中提交用户的令牌。对于每个请求，Verified Permissions 将令牌中的声明与策略进行比较。Verified Permissions 策略类似于 Amazon中的 IAM policy。该策略会声明主体、资源 和操作。如果您对于 `Allow` 的请求与允许的操作匹配，但与显式 `Deny` 操作不匹配，则 Verified Permissions 将对您的请求进行响应；否则，它将以 `Deny` 进行响应。有关更多信息，请参阅《Amazon Verified Permissions 用户指南》中的 Amazon Verified Permissions 策略。

自定义令牌

要更改、添加和删除您想要向 Verified Permissions 提供的用户声明，请使用令牌生成前 Lambda 触发器自定义访问令牌和身份令牌中的内容。使用令牌生成前触发器，您可以在令牌中添加和修改声明。例如，您可以在数据库中查询其他用户属性，并将这些属性编码为您的 ID 令牌。

> **Note**
>
> 由于 Verified Permissions 处理声明的方式，请勿在令牌生成前函数中添加名为 `cognito`、`dev` 或 `custom` 的声明。如果您提供的这些保留的声明前缀不是采用以冒号分隔的格式（例如 `cognito:username`），而是采用完整的声明名称，则您的授权请求会失败。

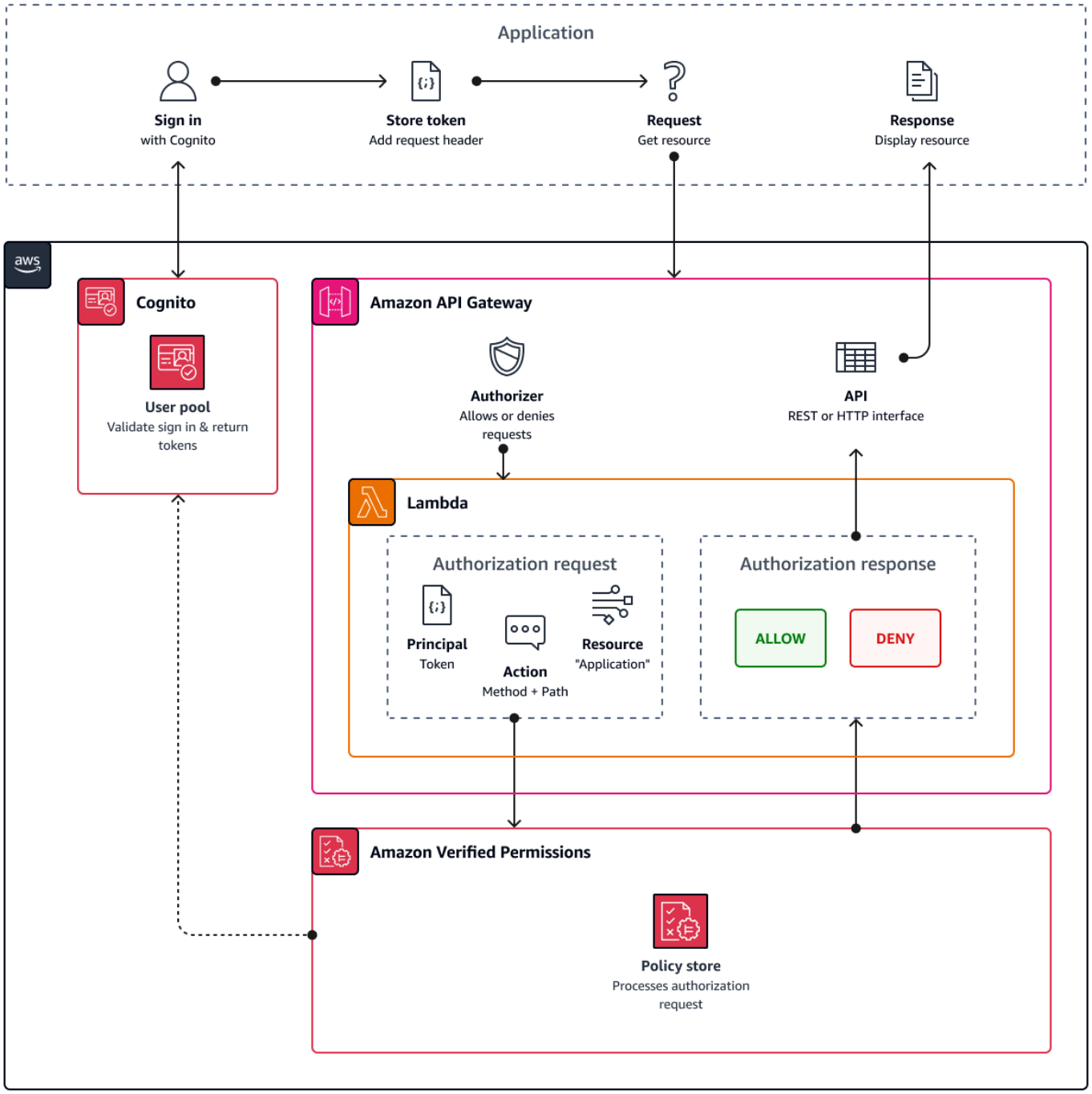
其他资源

- 将 Amazon Cognito 令牌映射到 Verified Permissions 架构
- APIs 使用亚马逊验证权限和亚马逊 Cognito 授权 API Gateway
- 研讨会：使用 Amazon Cognito 进行身份验证和授权以及经过验证的权限

使用 Verified Permissions 进行 API 授权

您的身份证或访问令牌可以使用经过验证的权限授权向后端 Amazon API Gateway REST APIs 发出的请求。您可以创建一个策略存储，其中包含指向您的用户池和 API 的直接链接。通过使用 API Gateway 设置和身份源启动选项，“已验证权限” 将用户池身份源添加到策略存储中，并向 API 添加一个 Lambda 授权方。当您的应用程序将用户池持有者令牌传递给 API 时，Lambda 授权方会调用 Verified Permissions。授权方将令牌作为主体来传递，将请求路径和方法作为操作来传递。

下图说明了使用 Verified Permissions 的 API Gateway API 的授权流程。有关详细明细，请参阅《Amazon Verified Permissions 用户指南》中与 API 相关的策略存储。



Verified Permissions 围绕用户池组构造 API 授权。由于 ID 和访问令牌都包含`cognito:groups`声明，因此您的策略存储可以在各种应用程序上下文 APIs 中为您管理基于角色的访问控制 (RBAC)。

选择策略存储设置

在策略存储上配置身份源时，必须选择是要处理访问令牌还是 ID 令牌。此决定对策略引擎的运作方式非常重要。ID 令牌包含用户属性。访问令牌包含用户访问控制信息：OAuth 范围。尽管两种令牌类型都有组成员资格信息，但我们通常建议使用 Verified Permissions 策略存储来进行基于角色的访问控制时使用访问令牌。访问令牌为组成员资格增加了范围，有助于授权决策。访问令牌中的声明成为授权请求中的上下文。

在将用户池配置为身份源时，还必须配置用户和组实体类型。实体类型是您可以在 Verified Permissions 策略中引用的主体、操作和资源标识符。策略存储中的实体可以具有成员关系，其中一个实体可以是父实体的成员。通过成员资格，您可以引用主体组、操作组和资源组。对于用户池组，您指定的用户实体类型必须是该组实体类型的成员。当您在 Verified Permissions 控制台中设置 API 相关策略存储或遵循引导式设置时，您的策略存储会自动建立这种父成员关系。

ID 令牌可以将 RBAC 与基于属性的访问权限控制（ABAC）结合使用。创建 API 相关策略存储后，您可以使用用户属性和组成员资格来增强策略。ID 令牌中的属性声明成为授权请求中的主体属性。您的策略可以根据主体属性作出授权决定。

您也可以将策略存储配置为接受与您提供的可接受应用程序客户端列表相匹配且具有 `aud` 或 `client_id` 声明的令牌。

基于角色的 API 授权的示例策略

以下示例策略是通过为示PetStore例 REST API 设置已验证权限策略存储而创建的。

```
permit(
  principal in PetStore::UserGroup::"us-east-1_EXAMPLE|MyGroup",
  action in [ PetStore::Action::"get /pets", PetStore::Action::"get /pets/{petId}" ],
  resource
  );
```

在以下情况下，Verified Permissions 返回对您的应用程序的授权请求的 `Allow` 决定：

1. 您的应用程序在 `Authorization` 标头中传递了 ID 令牌或访问令牌作为持有者令牌。
2. 您的应用程序传递一个具有 `cognito:groups` 声明的令牌，其中包含字符串 `MyGroup`。
3. 例如，您的应用程序向 `https://myapi.example.com/pets` 或 `https://myapi.example.com/pets/scrappy` 发出 `HTTP GET` 请求。

Amazon Cognito 用户的示例策略

您的用户池还可以在 API 请求以外的条件下生成向 Verified Permissions 发出的授权请求。您可以将应用程序中的任何访问控制决策提交到策略存储。例如，您可以在任何请求传输到网络之前，通过基于属性的访问控制来增强 Amazon DynamoDB 或 Amazon S3 的安全性，从而减少配额使用量。

以下示例使用 Cedar 策略语言，以允许通过一个用户群体应用程序客户端进行身份验证的财务用户读写 `example_image.png`。John 是应用程序中的用户，他从应用程序客户端接收 ID 令牌，并在 GET 请求中将其传递到需要授权的 URL `https://example.com/images/example_image.png`。John 的 ID 令牌拥有用户群体应用程序客户端 ID `1234567890example` 的 `aud` 声明。令牌生成前 Lambda 函数还插入了一个新声明 `costCenter`，对于 John 来说，值为 `Finance1234`。

```
permit (
   principal,
   actions in [ExampleCorp::Action::"readFile", "writeFile"],
   resource == ExampleCorp::Photo::"example_image.png"
)
when {
   principal.aud == "1234567890example" &&
   principal.custom.costCenter like "Finance*"
};
```

以下请求正文会导致 `Allow` 响应。

```
{
   "accesstoken": "[John's ID token]",
   "action": {
      "actionId": "readFile",
      "actionType": "Action"
   },
   "resource": {
      "entityId": "example_image.png",
      "entityType": "Photo"
   }
}
```

当您要在 Verified Permissions 策略中指定主体时，请使用以下格式：

```
permit (
   principal == [Namespace]::[Entity]::"[user pool ID]|[user sub]",
```

```
   action,
   resource
);
```

以下是 ID 为 `us-east-1_Example`（带有子项）的用户池中的用户或用户 ID 为 `973db890-092c-49e4-a9d0-912a4c0a20c7` 的用户的示例主体。

```
principal == ExampleCorp::User::"us-east-1_Example|973db890-092c-49e4-
a9d0-912a4c0a20c7",
```

当您要在 Verified Permissions 策略中指定用户组时，请使用以下格式：

```
permit (
   principal in [Namespace]::[Group Entity]::"[Group name]",
   action,
   resource
);
```

基于属性的访问控制

为您的应用程序提供经过验证的权限授权，以及用于 Amazon 凭证的 Amazon Cognito 身份池的访问控制属性功能，都是基于属性的访问控制 (ABAC) 的形式。以下是 Verified Permissions 和 Amazon Cognito ABAC 的功能比较。在 ABAC 中，系统检查实体的属性，并根据您定义的条件做出授权决策。

服务	流程	结果
Amazon Verified Permissions	根据对用户池 JWT 的分析返回`Allow`或`Deny`决策。	根据 Cedar 策略评估，应用程序资源的访问成功或失败。
Amazon Cognito 身份池（用于访问控制的属性）	根据用户的属性为其分配会话标签。IAM 策略条件可以检查标签`Allow`或`Deny`用户访问权限 Amazon Web Services 服务。	带有 IAM 角色临时 Amazon 证书的标记会话。

使用 Amazon Cognito 的代码示例 Amazon SDKs

以下代码示例展示了如何将 Amazon Cognito 与 Amazon 软件开发套件 (SDK) 一起使用。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用 Amazon Cognito 身份的代码示例 Amazon SDKs
  - 使用 Amazon Cognito 身份的基本示例 Amazon SDKs
    - 使用 Amazon Cognito 身份执行的操作 Amazon SDKs
      - CreateIdentityPool与 Amazon SDK 或 CLI 配合使用
      - DeleteIdentityPool与 Amazon SDK 或 CLI 配合使用
      - 将 DescribeIdentityPool 与 CLI 配合使用
      - GetCredentialsForIdentity与 Amazon SDK 一起使用
      - 将 GetIdentityPoolRoles 与 CLI 配合使用
      - ListIdentityPools与 Amazon SDK 或 CLI 配合使用
      - 将 SetIdentityPoolRoles 与 CLI 配合使用
      - 将 UpdateIdentityPool 与 CLI 配合使用
  - 使用 Amazon Cognito 身份的场景 Amazon SDKs
    - 创建 Amazon Textract 浏览器应用程序
- 使用 Amazon Cognito 身份提供商的代码示例 Amazon SDKs
  - 亚马逊 Cognito 身份提供商使用的基本示例 Amazon SDKs
    - 开始使用 Amazon Cognito
    - 亚马逊 Cognito 身份提供商使用的操作 Amazon SDKs
      - AdminCreateUser与 Amazon SDK 或 CLI 配合使用
      - AdminGetUser与 Amazon SDK 或 CLI 配合使用
      - AdminInitiateAuth与 Amazon SDK 或 CLI 配合使用
      - AdminRespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
      - AdminSetUserPassword与 Amazon SDK 或 CLI 配合使用
      - AssociateSoftwareToken与 Amazon SDK 或 CLI 配合使用
      - ConfirmDevice与 Amazon SDK 或 CLI 配合使用

- ConfirmForgotPassword与 Amazon SDK 或 CLI 配合使用
- ConfirmSignUp与 Amazon SDK 或 CLI 配合使用
- CreateUserPool与 Amazon SDK 或 CLI 配合使用
- CreateUserPoolClient与 Amazon SDK 或 CLI 配合使用
- DeleteUser与 Amazon SDK 或 CLI 配合使用
- ForgotPassword与 Amazon SDK 或 CLI 配合使用
- InitiateAuth与 Amazon SDK 或 CLI 配合使用
- ListUserPools与 Amazon SDK 或 CLI 配合使用
- ListUsers与 Amazon SDK 或 CLI 配合使用
- ResendConfirmationCode与 Amazon SDK 或 CLI 配合使用
- RespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
- SignUp与 Amazon SDK 或 CLI 配合使用
- UpdateUserPool与 Amazon SDK 或 CLI 配合使用
- VerifySoftwareToken与 Amazon SDK 或 CLI 配合使用

- 亚马逊 Cognito 身份提供商使用的场景 Amazon SDKs
  - 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 Amazon
  - 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 Amazon
  - 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 Amazon
  - 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 Amazon

- 使用 Amazon Cognito 同步的代码示例 Amazon SDKs
  - 使用 Amazon Cognito 同步的基本示例 Amazon SDKs
    - 使用 Amazon Cognito 同步的操作 Amazon SDKs
      - 与 Amazon SDK ListIdentityPoolUsage 配合使用

使用 Amazon Cognito 身份的代码示例 Amazon SDKs

以下代码示例展示了如何将 Amazon Cognito Identity 与 Amazon 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务服务结合来完成特定任务的代码示例。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用 Amazon Cognito 身份的基本示例 Amazon SDKs
  - 使用 Amazon Cognito 身份执行的操作 Amazon SDKs
    - CreateIdentityPool与 Amazon SDK 或 CLI 配合使用
    - DeleteIdentityPool与 Amazon SDK 或 CLI 配合使用
    - 将 DescribeIdentityPool 与 CLI 配合使用
    - GetCredentialsForIdentity与 Amazon SDK 一起使用
    - 将 GetIdentityPoolRoles 与 CLI 配合使用
    - ListIdentityPools与 Amazon SDK 或 CLI 配合使用
    - 将 SetIdentityPoolRoles 与 CLI 配合使用
    - 将 UpdateIdentityPool 与 CLI 配合使用
- 使用 Amazon Cognito 身份的场景 Amazon SDKs
  - 创建 Amazon Textract 浏览器应用程序

使用 Amazon Cognito 身份的基本示例 Amazon SDKs

以下代码示例展示了如何使用 Amazon Cognito Identity 的基础知识。 Amazon SDKs

示例

- 使用 Amazon Cognito 身份执行的操作 Amazon SDKs
  - CreateIdentityPool与 Amazon SDK 或 CLI 配合使用
  - DeleteIdentityPool与 Amazon SDK 或 CLI 配合使用
  - 将 DescribeIdentityPool 与 CLI 配合使用
  - GetCredentialsForIdentity与 Amazon SDK 一起使用
  - 将 GetIdentityPoolRoles 与 CLI 配合使用
  - ListIdentityPools与 Amazon SDK 或 CLI 配合使用
  - 将 SetIdentityPoolRoles 与 CLI 配合使用

- 将 UpdateIdentityPool 与 CLI 配合使用

使用 Amazon Cognito 身份执行的操作 Amazon SDKs

以下代码示例演示了如何使用执行单个 Amazon Cognito 身份操作。 Amazon SDKs每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon Cognito 身份 API，是必须在上下文中运行的较大型程序的代码节选。您可以在使用 Amazon Cognito 身份的场景 Amazon SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 身份 API 参考。

示例

- CreateIdentityPool与 Amazon SDK 或 CLI 配合使用
- DeleteIdentityPool与 Amazon SDK 或 CLI 配合使用
- 将 DescribeIdentityPool 与 CLI 配合使用
- GetCredentialsForIdentity与 Amazon SDK 一起使用
- 将 GetIdentityPoolRoles 与 CLI 配合使用
- ListIdentityPools与 Amazon SDK 或 CLI 配合使用
- 将 SetIdentityPoolRoles 与 CLI 配合使用
- 将 UpdateIdentityPool 与 CLI 配合使用

`CreateIdentityPool`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateIdentityPool`。

CLI

Amazon CLI

创建带有 Cognito 身份池提供者的身份池

此示例创建了一个名为的身份池 MyIdentityPool。它有一个 Cognito 身份池提供者。不允许使用未经身份验证的身份。

命令:

```
aws cognito-identity create-identity-pool --identity-pool-
name MyIdentityPool --no-allow-unauthenticated-identities --cognito-
```

```
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_aaaaaaaaa",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

输出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateIdentityPool中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.CreateIdentityPoolResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
```

```
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateIdentityPool {
    public static void main(String[] args) {
        final String usage = """
                Usage:
                    <identityPoolName>\s

                Where:
                    identityPoolName - The name to give your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoolName = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String identityPoolId = createIdPool(cognitoClient, identityPoolName);
        System.out.println("Unity pool ID " + identityPoolId);
        cognitoClient.close();
    }

    public static String createIdPool(CognitoIdentityClient cognitoClient, String
 identityPoolName) {
        try {
            CreateIdentityPoolRequest poolRequest =
 CreateIdentityPoolRequest.builder()
                    .allowUnauthenticatedIdentities(false)
                    .identityPoolName(identityPoolName)
                    .build();

            CreateIdentityPoolResponse response =
 cognitoClient.createIdentityPool(poolRequest);
```

```
            return response.identityPoolId();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateIdentityPool中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：创建允许未经身份验证的身份的新身份池。

```
New-CGIIdentityPool -AllowUnauthenticatedIdentities $true -IdentityPoolName
 CommonTests13
```

输出：

```
LoggedAt                       : 8/12/2015 4:56:07 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3
IdentityPoolName               : CommonTests13
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 136
HttpStatusCode                 : OK
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) CreateIdentityPool中的。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentity


    /// Create a new identity pool and return its ID.
    ///
    /// - Parameters:
    ///     - name: The name to give the new identity pool.
    ///
    /// - Returns: A string containing the newly created pool's ID, or `nil`
    ///   if an error occurred.
    ///
    func createIdentityPool(name: String) async throws -> String? {
        do {
            let cognitoInputCall = CreateIdentityPoolInput(developerProviderName:
 "com.exampleco.CognitoIdentityDemo",
                                                           identityPoolName:
 name)

            let result = try await
 cognitoIdentityClient.createIdentityPool(input: cognitoInputCall)
            guard let poolId = result.identityPoolId else {
                return nil
            }

            return poolId
        } catch {
            print("ERROR: createIdentityPool:", dump(error))
            throw error
        }
    }
```

- 有关更多信息，请参阅 Amazon SDK for Swift 开发人员指南。
- 如需了解 API 的详细信息，请参阅适用CreateIdentityPool于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`DeleteIdentityPool`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `DeleteIdentityPool`。

CLI

Amazon CLI

删除身份池

以下 `delete-identity-pool` 示例删除指定身份池。

命令:

```
aws cognito-identity delete-identity-pool \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

此命令不生成任何输出。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteIdentityPool中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.auth.credentials.ProfileCredentialsProvider;
import software.amazon.awssdk.awscore.exception.AwsServiceException;
import software.amazon.awssdk.regions.Region;
```

```
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.DeleteIdentityPoolRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class DeleteIdentityPool {

    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <identityPoolId>\s

                Where:
                    identityPoolId - The Id value of your identity pool.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityPoold = args[0];
        CognitoIdentityClient cognitoIdClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .credentialsProvider(ProfileCredentialsProvider.create())
                .build();

        deleteIdPool(cognitoIdClient, identityPoold);
        cognitoIdClient.close();
    }

    public static void deleteIdPool(CognitoIdentityClient cognitoIdClient, String
 identityPoold) {
        try {
```

```
            DeleteIdentityPoolRequest identityPoolRequest =
 DeleteIdentityPoolRequest.builder()
                    .identityPoolId(identityPoold)
                    .build();

            cognitoIdClient.deleteIdentityPool(identityPoolRequest);
            System.out.println("Done");

        } catch (AwsServiceException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考DeleteIdentityPool中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：删除特定身份池。

```
Remove-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) DeleteIdentityPool中的。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentity


    /// Delete the specified identity pool.
    ///
    /// - Parameters:
    ///   - id: The ID of the identity pool to delete.
    ///
    func deleteIdentityPool(id: String) async throws {
        do {
            let input = DeleteIdentityPoolInput(
                identityPoolId: id
            )

            _ = try await cognitoIdentityClient.deleteIdentityPool(input: input)
        } catch {
            print("ERROR: deleteIdentityPool:", dump(error))
            throw error
        }
    }
```

- 有关更多信息，请参阅 Amazon SDK for Swift 开发人员指南。
- 如需了解 API 的详细信息，请参阅适用DeleteIdentityPool于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`DescribeIdentityPool`** 与 CLI 配合使用

以下代码示例演示如何使用 `DescribeIdentityPool`。

CLI

Amazon CLI

描述身份池

此示例描述身份池。

命令:

```
aws cognito-identity describe-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111"
```

输出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DescribeIdentityPool中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：按身份池的 ID 检索有关该身份池的信息。

```
Get-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

输出：

```
LoggedAt                       : 8/12/2015 4:29:40 PM
AllowUnauthenticatedIdentities : True
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : CommonTests1
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 142
```

```
HttpStatusCode                 : OK
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) DescribeIdentityPool中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`GetCredentialsForIdentity`与 Amazon SDK 一起使用

以下代码示例演示了如何使用 `GetCredentialsForIdentity`。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.GetCredentialsForIdentityResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class GetIdentityCredentials {
```

```
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <identityId>\s

            Where:
                identityId - The Id of an existing identity in the format
REGION:GUID.
            """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String identityId = args[0];
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        getCredsForIdentity(cognitoClient, identityId);
        cognitoClient.close();
    }

    public static void getCredsForIdentity(CognitoIdentityClient cognitoClient,
String identityId) {
        try {
            GetCredentialsForIdentityRequest getCredentialsForIdentityRequest =
GetCredentialsForIdentityRequest
                    .builder()
                    .identityId(identityId)
                    .build();

            GetCredentialsForIdentityResponse response = cognitoClient
                    .getCredentialsForIdentity(getCredentialsForIdentityRequest);
            System.out.println(
                    "Identity ID " + response.identityId() + ", Access key ID " +
response.credentials().accessKeyId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考GetCredentialsForIdentity中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **`GetIdentityPoolRoles`** 与 CLI 配合使用

以下代码示例演示如何使用 `GetIdentityPoolRoles`。

CLI

Amazon CLI

获取身份池角色

此示例获取身份池角色。

命令:

```
aws cognito-identity get-identity-pool-roles --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111"
```

输出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "Roles": {
      "authenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolAuth_Role",
      "unauthenticated": "arn:aws:iam::111111111111:role/Cognito_MyIdentityPoolUnauth_Role"
  }
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考GetIdentityPoolRoles中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：获取有关特定身份池的角色的信息。

```
Get-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1
```

输出：

```
LoggedAt         : 8/12/2015 4:33:51 PM
IdentityPoolId   : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
Roles            : {[unauthenticated, arn:aws:iam::123456789012:role/
CommonTests1Role]}
ResponseMetadata : Amazon.Runtime.ResponseMetadata
ContentLength    : 165
HttpStatusCode   : OK
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) GetIdentityPoolRoles中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListIdentityPools`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListIdentityPools`。

CLI

Amazon CLI

列出身份池

此示例列出身份池。最多可列出 20 个身份。

命令:

```
aws cognito-identity list-identity-pools --max-results 20
```

输出：

```
{
  "IdentityPools": [
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "MyIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "AnotherIdentityPool"
      },
      {
          "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
          "IdentityPoolName": "IdentityPoolRegionA"
      }
  ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListIdentityPools中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import software.amazon.awssdk.services.cognitoidentity.CognitoIdentityClient;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsRequest;
import
 software.amazon.awssdk.services.cognitoidentity.model.ListIdentityPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
```

```
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListIdentityPools {
    public static void main(String[] args) {
        CognitoIdentityClient cognitoClient = CognitoIdentityClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listIdPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listIdPools(CognitoIdentityClient cognitoClient) {
        try {
            ListIdentityPoolsRequest poolsRequest =
 ListIdentityPoolsRequest.builder()
                    .maxResults(15)
                    .build();

            ListIdentityPoolsResponse response =
 cognitoClient.listIdentityPools(poolsRequest);
            response.identityPools().forEach(pool -> {
                System.out.println("Pool ID: " + pool.identityPoolId());
                System.out.println("Pool name: " + pool.identityPoolName());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListIdentityPools中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：检索现有身份池的列表。

```
Get-CGIIdentityPoolList
```

输出：

```
IdentityPoolId
 IdentityPoolName
--------------
 ----------------
us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1                    CommonTests1
us-east-1:118d242d-204e-4b88-b803-EXAMPLEGUID2                    Tests2
us-east-1:15d49393-ab16-431a-b26e-EXAMPLEGUID3                    CommonTests13
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) ListIdentityPools中的。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentity


    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned.
    ///
    /// - Returns: A string containing the ID of the specified identity pool
```

```
    ///   or `nil` on error or if not found.
    ///
    func getIdentityPoolID(name: String) async throws -> String? {
        let listPoolsInput = ListIdentityPoolsInput(maxResults: 25)
        // Use "Paginated" to get all the objects.
        // This lets the SDK handle the 'nextToken' field in
 "ListIdentityPoolsOutput".
        let pages = cognitoIdentityClient.listIdentityPoolsPaginated(input:
 listPoolsInput)

        do {
            for try await page in pages {
                guard let identityPools = page.identityPools else {
                    print("ERROR: listIdentityPoolsPaginated returned nil
 contents.")
                    continue
                }

                /// Read pages of identity pools from Cognito until one is found
                /// whose name matches the one specified in the `name` parameter.
                /// Return the matching pool's ID.

                for pool in identityPools {
                    if pool.identityPoolName == name {
                        return pool.identityPoolId!
                    }
                }
            }
        } catch {
            print("ERROR: getIdentityPoolID:", dump(error))
            throw error
        }

        return nil
    }
```

获取现有身份池的 ID，如果它不存在，则创建它。

```
import AWSCognitoIdentity
```

```
    /// Return the ID of the identity pool with the specified name.
    ///
    /// - Parameters:
    ///   - name: The name of the identity pool whose ID should be returned
    ///
    /// - Returns: A string containing the ID of the specified identity pool.
    ///   Returns `nil` if there's an error or if the pool isn't found.
    ///
    public func getOrCreateIdentityPoolID(name: String) async throws -> String? {
        // See if the pool already exists. If it doesn't, create it.

        do {
            guard let poolId = try await getIdentityPoolID(name: name) else {
                return try await createIdentityPool(name: name)
            }

            return poolId
        } catch {
            print("ERROR: getOrCreateIdentityPoolID:", dump(error))
            throw error
        }
    }
```

- 有关更多信息，请参阅 Amazon SDK for Swift 开发人员指南。
- 如需了解 API 的详细信息，请参阅适用ListIdentityPools于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **SetIdentityPoolRoles** 与 CLI 配合使用

以下代码示例演示如何使用 `SetIdentityPoolRoles`。

CLI

Amazon CLI

设置身份池角色

以下 `set-identity-pool-roles` 示例设置身份池角色。

```
aws cognito-identity set-identity-pool-roles \
    --identity-pool-id "us-west-2:11111111-1111-1111-1111-111111111111" \
    --roles authenticated="arn:aws:iam::111111111111:role/
Cognito_MyIdentityPoolAuth_Role"
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SetIdentityPoolRoles中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：将特定的身份池配置为具有未经身份验证的 IAM 角色。

```
Set-CGIIdentityPoolRole -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -Role @{ "unauthenticated" = "arn:aws:iam::123456789012:role/
CommonTests1Role" }
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) SetIdentityPoolRoles中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

将 **UpdateIdentityPool** 与 CLI 配合使用

以下代码示例演示如何使用 `UpdateIdentityPool`。

CLI

Amazon CLI

更新身份池

此示例更新身份池。它将名称设置为 MyIdentityPool。它添加 Cognito 作为身份池提供者。它不允许使用未经身份验证的身份。

命令:

```
aws cognito-identity update-identity-pool --identity-pool-id "us-
west-2:11111111-1111-1111-1111-111111111111" --identity-pool-
name "MyIdentityPool" --no-allow-unauthenticated-identities --cognito-
```

```
identity-providers ProviderName="cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",ClientId="3n4b5urk1ft4fl3mg5e62d9ado",ServerSideTokenCheck=false
```

输出：

```
{
  "IdentityPoolId": "us-west-2:11111111-1111-1111-1111-111111111111",
  "IdentityPoolName": "MyIdentityPool",
  "AllowUnauthenticatedIdentities": false,
  "CognitoIdentityProviders": [
      {
          "ProviderName": "cognito-idp.us-west-2.amazonaws.com/us-
west-2_111111111",
          "ClientId": "3n4b5urk1ft4fl3mg5e62d9ado",
          "ServerSideTokenCheck": false
      }
  ]
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考UpdateIdentityPool中的。

PowerShell

适用于 PowerShell V4 的工具

示例 1：更新某些身份池属性，在本例中为身份池的名称。

```
Update-CGIIdentityPool -IdentityPoolId us-east-1:0de2af35-2988-4d0b-b22d-
EXAMPLEGUID1 -IdentityPoolName NewPoolName
```

输出：

```
LoggedAt                       : 8/12/2015 4:53:33 PM
AllowUnauthenticatedIdentities : False
DeveloperProviderName          :
IdentityPoolId                 : us-east-1:0de2af35-2988-4d0b-b22d-EXAMPLEGUID1
IdentityPoolName               : NewPoolName
OpenIdConnectProviderARNs      : {}
SupportedLoginProviders        : {}
ResponseMetadata               : Amazon.Runtime.ResponseMetadata
ContentLength                  : 135
```

```
HttpStatusCode                 : OK
```

- 有关 API 的详细信息，请参阅 Amazon Tools for PowerShell Cmdlet 参考 (V 4) UpdateIdentityPool中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Cognito 身份的场景 Amazon SDKs

以下代码示例向您展示了如何使用在 Amazon Cognito Identity 中实现常见场景。 Amazon SDKs这些场景向您展示了如何通过在 Amazon Cognito Identity 中调用多个函数或结合其他 Amazon Web Services 服务来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 创建 Amazon Textract 浏览器应用程序

创建 Amazon Textract 浏览器应用程序

以下代码示例展示如何通过交互式应用程序探索 Amazon Textract 输出。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

演示如何使用 适用于 JavaScript 的 Amazon SDK 来构建 React 应用程序，该应用程序使用 Amazon Textract 从文档图像中提取数据并将其显示在交互式网页中。此示例在 Web 浏览器中运行，需要经过身份验证的 Amazon Cognito 身份才能获得凭证。它使用 Amazon Simple Storage Service（Amazon S3）进行存储；对于通知，它将轮询订阅 Amazon Simple Notification Service（Amazon SNS）主题的 Amazon Simple Queue Service（Amazon SQS）队列。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Cognito Identity

- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

Python

适用于 Python 的 SDK（Boto3）

演示如何 适用于 Python (Boto3) 的 Amazon SDK 与 Amazon Textract 配合使用来检测文档图像中的文本、表单和表格元素。输入图像和 Amazon Textract 输出在 Tkinter 应用程序中显示，该应用程序可让您探索检测到的元素。

- 将文档图像提交到 Amazon Textract 并探索检测到的元素的输出。
- 将图像直接提交到 Amazon Textract，或通过 Amazon Simple Storage Service（Amazon S3）桶提交图像。
- 使用异步 APIs 启动任务，该任务在任务完成时向亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 主题发布通知。
- 轮询 Amazon Simple Queue Service (Amazon SQS) 队列，以获取任务完成消息并显示结果。

有关如何设置和运行的完整源代码和说明，请参阅上的完整示例GitHub。

本示例中使用的服务

- Amazon Cognito Identity
- Amazon S3
- Amazon SNS
- Amazon SQS
- Amazon Textract

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Cognito 身份提供商的代码示例 Amazon SDKs

以下代码示例展示了如何将 Amazon Cognito 身份提供程序与 Amazon 软件开发套件 (SDK) 配合使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

场景是向您展示如何通过在一个服务中调用多个函数或与其他 Amazon Web Services 服务服务结合来完成特定任务的代码示例。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

开始使用

开始使用 Amazon Cognito

以下代码示例显示如何开始使用 Amazon Cognito。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")
```

```
# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.

     # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                     # and set the proper subdirectory to the
 executables' location.

     AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_cognito.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>
```

```
/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
 cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
 listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
 listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
 listUserPoolsOutcome.GetResult().GetUserPools()) {
```

```
                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
 listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListUserPools中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
package main

import (
 "context"
 "fmt"
```

```
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListUserPools中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
```

```
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListUserPools中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";
```

```
const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListUserPools中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
```

```
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- 有关 API 的详细信息，请参阅适用ListUserPools于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
```

```
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考ListUserPools中的。

代码示例

- 亚马逊 Cognito 身份提供商使用的基本示例 Amazon SDKs
  - 开始使用 Amazon Cognito
  - 亚马逊 Cognito 身份提供商使用的操作 Amazon SDKs
    - AdminCreateUser与 Amazon SDK 或 CLI 配合使用
    - AdminGetUser与 Amazon SDK 或 CLI 配合使用

- AdminInitiateAuth与 Amazon SDK 或 CLI 配合使用
- AdminRespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
- AdminSetUserPassword与 Amazon SDK 或 CLI 配合使用
- AssociateSoftwareToken与 Amazon SDK 或 CLI 配合使用
- ConfirmDevice与 Amazon SDK 或 CLI 配合使用
- ConfirmForgotPassword与 Amazon SDK 或 CLI 配合使用
- ConfirmSignUp与 Amazon SDK 或 CLI 配合使用
- CreateUserPool与 Amazon SDK 或 CLI 配合使用
- CreateUserPoolClient与 Amazon SDK 或 CLI 配合使用
- DeleteUser与 Amazon SDK 或 CLI 配合使用
- ForgotPassword与 Amazon SDK 或 CLI 配合使用
- InitiateAuth与 Amazon SDK 或 CLI 配合使用
- ListUserPools与 Amazon SDK 或 CLI 配合使用
- ListUsers与 Amazon SDK 或 CLI 配合使用
- ResendConfirmationCode与 Amazon SDK 或 CLI 配合使用
- RespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
- SignUp与 Amazon SDK 或 CLI 配合使用
- UpdateUserPool与 Amazon SDK 或 CLI 配合使用
- VerifySoftwareToken与 Amazon SDK 或 CLI 配合使用

- 亚马逊 Cognito 身份提供商使用的场景 Amazon SDKs
  - 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 Amazon
  - 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 Amazon
  - 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 Amazon
  - 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 Amazon

亚马逊 Cognito 身份提供商使用的基本示例 Amazon SDKs

以下代码示例展示了如何使用 Amazon Cognito 身份提供商的基础知识。 Amazon SDKs

示例

- 开始使用 Amazon Cognito

- 亚马逊 Cognito 身份提供商使用的操作 Amazon SDKs
  - AdminCreateUser与 Amazon SDK 或 CLI 配合使用
  - AdminGetUser与 Amazon SDK 或 CLI 配合使用
  - AdminInitiateAuth与 Amazon SDK 或 CLI 配合使用
  - AdminRespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
  - AdminSetUserPassword与 Amazon SDK 或 CLI 配合使用
  - AssociateSoftwareToken与 Amazon SDK 或 CLI 配合使用
  - ConfirmDevice与 Amazon SDK 或 CLI 配合使用
  - ConfirmForgotPassword与 Amazon SDK 或 CLI 配合使用
  - ConfirmSignUp与 Amazon SDK 或 CLI 配合使用
  - CreateUserPool与 Amazon SDK 或 CLI 配合使用
  - CreateUserPoolClient与 Amazon SDK 或 CLI 配合使用
  - DeleteUser与 Amazon SDK 或 CLI 配合使用
  - ForgotPassword与 Amazon SDK 或 CLI 配合使用
  - InitiateAuth与 Amazon SDK 或 CLI 配合使用
  - ListUserPools与 Amazon SDK 或 CLI 配合使用
  - ListUsers与 Amazon SDK 或 CLI 配合使用
  - ResendConfirmationCode与 Amazon SDK 或 CLI 配合使用
  - RespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
  - SignUp与 Amazon SDK 或 CLI 配合使用
  - UpdateUserPool与 Amazon SDK 或 CLI 配合使用
  - VerifySoftwareToken与 Amazon SDK 或 CLI 配合使用

开始使用 Amazon Cognito

以下代码示例展示了如何开始使用 Amazon Cognito。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

CMakeLists.txt CMake 文件的代码。

```
# Set the minimum required version of CMake for this project.
cmake_minimum_required(VERSION 3.13)

# Set the AWS service components used by this project.
set(SERVICE_COMPONENTS cognito-idp)

# Set this project's name.
project("hello_cognito")

# Set the C++ standard to use to build this target.
# At least C++ 11 is required for the AWS SDK for C++.
set(CMAKE_CXX_STANDARD 11)

# Use the MSVC variable to determine if this is a Windows build.
set(WINDOWS_BUILD ${MSVC})

if (WINDOWS_BUILD) # Set the location where CMake can find the installed
 libraries for the AWS SDK.
    string(REPLACE ";" "/aws-cpp-sdk-all;" SYSTEM_MODULE_PATH
 "${CMAKE_SYSTEM_PREFIX_PATH}/aws-cpp-sdk-all")
    list(APPEND CMAKE_PREFIX_PATH ${SYSTEM_MODULE_PATH})
endif ()

# Find the AWS SDK for C++ package.
find_package(AWSSDK REQUIRED COMPONENTS ${SERVICE_COMPONENTS})

if (WINDOWS_BUILD AND AWSSDK_INSTALL_AS_SHARED_LIBS)
     # Copy relevant AWS SDK for C++ libraries into the current binary directory
 for running and debugging.
```

```
    # set(BIN_SUB_DIR "/Debug") # If you are building from the command line, you
 may need to uncomment this
                                    # and set the proper subdirectory to the
 executables' location.

    AWSSDK_CPY_DYN_LIBS(SERVICE_COMPONENTS ""
 ${CMAKE_CURRENT_BINARY_DIR}${BIN_SUB_DIR})
endif ()

add_executable(${PROJECT_NAME}
        hello_cognito.cpp)

target_link_libraries(${PROJECT_NAME}
        ${AWSSDK_LINK_LIBRARIES})
```

hello_cognito.cpp 源文件的代码。

```
#include <aws/core/Aws.h>
#include <aws/cognito-idp/CognitoIdentityProviderClient.h>
#include <aws/cognito-idp/model/ListUserPoolsRequest.h>
#include <iostream>

/*
 *  A "Hello Cognito" starter application which initializes an Amazon Cognito
 client and lists the Amazon Cognito
 *  user pools.
 *
 *  main function
 *
 *  Usage: 'hello_cognito'
 *
 */

int main(int argc, char **argv) {
    Aws::SDKOptions options;
    // Optionally change the log level for debugging.
//   options.loggingOptions.logLevel = Utils::Logging::LogLevel::Debug;
    Aws::InitAPI(options); // Should only be called once.
    int result = 0;
    {
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

        Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
cognitoClient(clientConfig);

        Aws::String nextToken; // Used for pagination.
        std::vector<Aws::String> userPools;

        do {
            Aws::CognitoIdentityProvider::Model::ListUserPoolsRequest
listUserPoolsRequest;
            if (!nextToken.empty()) {
                listUserPoolsRequest.SetNextToken(nextToken);
            }

            Aws::CognitoIdentityProvider::Model::ListUserPoolsOutcome
listUserPoolsOutcome =
                    cognitoClient.ListUserPools(listUserPoolsRequest);

            if (listUserPoolsOutcome.IsSuccess()) {
                for (auto &userPool:
listUserPoolsOutcome.GetResult().GetUserPools()) {

                    userPools.push_back(userPool.GetName());
                }

                nextToken = listUserPoolsOutcome.GetResult().GetNextToken();
            } else {
                std::cerr << "ListUserPools error: " <<
listUserPoolsOutcome.GetError().GetMessage() << std::endl;
                result = 1;
                break;
            }


        } while (!nextToken.empty());
        std::cout << userPools.size() << " user pools found." << std::endl;
        for (auto &userPool: userPools) {
            std::cout << "   user pool: " << userPool << std::endl;
        }
    }

    Aws::ShutdownAPI(options); // Should only be called once.
    return result;
```

```
}
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ListUserPools中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  fmt.Println("Couldn't load default configuration. Have you set up your AWS
 account?")
  fmt.Println(err)
  return
```

```
 }
 cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
 fmt.Println("Let's list the user pools for your account.")
 var pools []types.UserPoolDescriptionType
 paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
  cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
 aws.Int32(10)})
 for paginator.HasMorePages() {
  output, err := paginator.NextPage(ctx)
  if err != nil {
   log.Printf("Couldn't get user pools. Here's why: %v\n", err)
  } else {
   pools = append(pools, output.UserPools...)
  }
 }
 if len(pools) == 0 {
  fmt.Println("You don't have any user pools!")
 } else {
  for _, pool := range pools {
   fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
  }
 }
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListUserPools中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
```

```
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListUserPools中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import {
  paginateListUserPools,
  CognitoIdentityProviderClient,
} from "@aws-sdk/client-cognito-identity-provider";

const client = new CognitoIdentityProviderClient({});

export const helloCognito = async () => {
  const paginator = paginateListUserPools({ client }, {});

  const userPoolNames = [];

  for await (const page of paginator) {
    const names = page.UserPools.map((pool) => pool.Name);
    userPoolNames.push(...names);
  }

  console.log("User pool names: ");
  console.log(userPoolNames.join("\n"));
  return userPoolNames;
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListUserPools中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import boto3

# Create a Cognito Identity Provider client
cognitoidp = boto3.client("cognito-idp")

# Initialize a paginator for the list_user_pools operation
paginator = cognitoidp.get_paginator("list_user_pools")

# Create a PageIterator from the paginator
page_iterator = paginator.paginate(MaxResults=10)

# Initialize variables for pagination
user_pools = []

# Handle pagination
for page in page_iterator:
    user_pools.extend(page.get("UserPools", []))

# Print the list of user pools
print("User Pools for the account:")
if user_pools:
    for pool in user_pools:
        print(f"Name: {pool['Name']}, ID: {pool['Id']}")
else:
    print("No user pools found.")
```

- 有关 API 的详细信息，请参阅适用ListUserPools于 Python 的Amazon SDK (Boto3) API 参考。

Ruby

适用于 Ruby 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
require 'aws-sdk-cognitoidentityprovider'
require 'logger'

# CognitoManager is a class responsible for managing AWS Cognito operations
# such as listing all user pools in the current AWS account.
class CognitoManager
  def initialize(client)
    @client = client
    @logger = Logger.new($stdout)
  end

  # Lists and prints all user pools associated with the AWS account.
  def list_user_pools
    paginator = @client.list_user_pools(max_results: 10)
    user_pools = []
    paginator.each_page do |page|
      user_pools.concat(page.user_pools)
    end

    if user_pools.empty?
      @logger.info('No Cognito user pools found.')
    else
      user_pools.each do |user_pool|
        @logger.info("User pool ID: #{user_pool.id}")
        @logger.info("User pool name: #{user_pool.name}")
        @logger.info("User pool status: #{user_pool.status}")
        @logger.info('---')
```

```
      end
    end
  end
end

if $PROGRAM_NAME == __FILE__
  cognito_client = Aws::CognitoIdentityProvider::Client.new
  manager = CognitoManager.new(cognito_client)
  manager.list_user_pools
end
```

- 有关 API 的详细信息，请参阅 适用于 Ruby 的 Amazon SDK API 参考ListUserPools中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

亚马逊 Cognito 身份提供商使用的操作 Amazon SDKs

以下代码示例演示了如何使用执行各个 Amazon Cognito 身份提供商操作。 Amazon SDKs每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

这些代码节选调用了 Amazon Cognito 身份提供者 API，是必须在上下文中运行的较大型程序的代码节选。您可以在亚马逊 Cognito 身份提供商使用的场景 Amazon SDKs 中结合上下文查看操作。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 身份提供者 API 参考。

示例

- AdminCreateUser与 Amazon SDK 或 CLI 配合使用
- AdminGetUser与 Amazon SDK 或 CLI 配合使用
- AdminInitiateAuth与 Amazon SDK 或 CLI 配合使用
- AdminRespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
- AdminSetUserPassword与 Amazon SDK 或 CLI 配合使用
- AssociateSoftwareToken与 Amazon SDK 或 CLI 配合使用
- ConfirmDevice与 Amazon SDK 或 CLI 配合使用
- ConfirmForgotPassword与 Amazon SDK 或 CLI 配合使用

- ConfirmSignUp与 Amazon SDK 或 CLI 配合使用
- CreateUserPool与 Amazon SDK 或 CLI 配合使用
- CreateUserPoolClient与 Amazon SDK 或 CLI 配合使用
- DeleteUser与 Amazon SDK 或 CLI 配合使用
- ForgotPassword与 Amazon SDK 或 CLI 配合使用
- InitiateAuth与 Amazon SDK 或 CLI 配合使用
- ListUserPools与 Amazon SDK 或 CLI 配合使用
- ListUsers与 Amazon SDK 或 CLI 配合使用
- ResendConfirmationCode与 Amazon SDK 或 CLI 配合使用
- RespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用
- SignUp与 Amazon SDK 或 CLI 配合使用
- UpdateUserPool与 Amazon SDK 或 CLI 配合使用
- VerifySoftwareToken与 Amazon SDK 或 CLI 配合使用

AdminCreateUser与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminCreateUser`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

CLI

Amazon CLI

创建用户

以下的 `admin-create-user` 示例创建具有指定设置电子邮件地址和电话号码的用户。

```
aws cognito-idp admin-create-user \
    --user-pool-id us-west-2_aaaaaaaaa \
    --username diego \
    --user-attributes Name=email,Value=diego@example.com
 Name=phone_number,Value="+15555551212" \
```

```
    --message-action SUPPRESS
```

输出：

```
{
    "User": {
        "Username": "diego",
        "Attributes": [
            {
                "Name": "sub",
                "Value": "7325c1de-b05b-4f84-b321-9adc6e61f4a2"
            },
            {
                "Name": "phone_number",
                "Value": "+15555551212"
            },
            {
                "Name": "email",
                "Value": "diego@example.com"
            }
        ],
        "UserCreateDate": 1548099495.428,
        "UserLastModifiedDate": 1548099495.428,
        "Enabled": true,
        "UserStatus": "FORCE_CHANGE_PASSWORD"
    }
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AdminCreateUser中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考AdminCreateUser中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

AdminGetUser与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminGetUser`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
```

```
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考AdminGetUser中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
```

```
    }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考AdminGetUser中的。

CLI

Amazon CLI

获取用户

此示例获取有关用户名 jane@example.com 的信息。

命令:

```
aws cognito-idp admin-get-user --user-pool-id us-west-2_aaaaaaaaa --
username jane@example.com
```

输出：

```
{
  "Username": "4320de44-2322-4620-999b-5e2e1c8df013",
  "Enabled": true,
  "UserStatus": "FORCE_CHANGE_PASSWORD",
  "UserCreateDate": 1548108509.537,
  "UserAttributes": [
      {
          "Name": "sub",
          "Value": "4320de44-2322-4620-999b-5e2e1c8df013"
      },
      {
          "Name": "email_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number_verified",
          "Value": "true"
      },
      {
          "Name": "phone_number",
          "Value": "+01115551212"
      },
```

```
        {
            "Name": "email",
            "Value": "jane@example.com"
        }
    ],
    "UserLastModifiedDate": 1548108509.537
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AdminGetUser中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static void getAdminUser(CognitoIdentityProviderClient
identityProviderClient, String userName,
           String poolId) {
       try {
           AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                   .username(userName)
                   .userPoolId(poolId)
                   .build();

           AdminGetUserResponse response =
identityProviderClient.adminGetUser(userRequest);
           System.out.println("User status " + response.userStatusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考AdminGetUser中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const adminGetUser = ({ userPoolId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminGetUserCommand({
    UserPoolId: userPoolId,
    Username: username,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考AdminGetUser中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
```

```
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}
```

- 有关 API 的详细信息，请参阅适用AdminGetUser于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
```

```
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- 有关 API 的详细信息，请参阅适用AdminGetUser于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
```

```
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
        } catch {
            return false
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用AdminGetUser于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminInitiateAuth`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminInitiateAuth`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Initiate an admin auth request.
```

```
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考AdminInitiateAuth中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
```

```
        // clientConfig.region = "us-east-1";

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

   Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
   request.SetClientId(clientID);
   request.SetUserPoolId(userPoolID);
   request.AddAuthParameters("USERNAME", userName);
   request.AddAuthParameters("PASSWORD", password);
   request.SetAuthFlow(

Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


   Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
           client.AdminInitiateAuth(request);

   if (outcome.IsSuccess()) {
       std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
       sessionResult = outcome.GetResult().GetSession();
   }
   else {
       std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                 << outcome.GetError().GetMessage()
                 << std::endl;
   }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考AdminInitiateAuth中的。

CLI

Amazon CLI

以管理员身份注册用户

以下 `admin-initiate-auth` 示例注册用户 diego@example.com。此示例还包括威胁防护和 Lambda 触发 ClientMetadata 器的元数据。已为用户配置 TOTP MFA，用户会收到质询，需要先提供来自其身份验证器应用程序的代码，之后才能完成身份验证。

```
aws cognito-idp admin-initiate-auth \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --auth-flow ADMIN_USER_PASSWORD_AUTH \
    --auth-parameters USERNAME=diego@example.com,PASSWORD="My@Example
$Password3!",SECRET_HASH=ExampleEncodedClientIdSecretAndUsername= \
    --context-data="{\"EncodedData\":\"abc123example\",\"HttpHeaders\":
[{\"headerName\":\"UserAgent\",\"headerValue\":\"Mozilla/5.0 (Windows NT
 6.1; Win64; x64; rv:47.0) Gecko/20100101 Firefox/47.0\"}],\"IpAddress\":
\"192.0.2.1\",\"ServerName\":\"example.com\",\"ServerPath\":\"/login\"}" \
    --client-metadata="{\"MyExampleKey\": \"MyExampleValue\"}"
```

输出：

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeExample...",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "MyAuthenticatorApp",
        "USER_ID_FOR_SRP": "diego@example.com"
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Admin authentication flow。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AdminInitiateAuth中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    public static AdminInitiateAuthResponse
 initiateAuth(CognitoIdentityProviderClient identityProviderClient,
            String clientId, String userName, String password, String userPoolId)
 {
```

```
        try {
            Map<String, String> authParameters = new HashMap<>();
            authParameters.put("USERNAME", userName);
            authParameters.put("PASSWORD", password);

            AdminInitiateAuthRequest authRequest =
 AdminInitiateAuthRequest.builder()
                    .clientId(clientId)
                    .userPoolId(userPoolId)
                    .authParameters(authParameters)
                    .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                    .build();

            AdminInitiateAuthResponse response =
 identityProviderClient.adminInitiateAuth(authRequest);
            System.out.println("Result Challenge is : " +
 response.challengeName());
            return response;

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考AdminInitiateAuth中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考AdminInitiateAuth中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
```

```
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
}
```

- 有关 API 的详细信息，请参阅适用AdminInitiateAuth于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret
```

```
    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
 not "
```

```
                        "configured for TOTP MFA. This example requires TOTP
 MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- 有关 API 的详细信息，请参阅适用AdminInitiateAuth于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
```

```
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用AdminInitiateAuth于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

AdminRespondToAuthChallenge与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 AdminRespondToAuthChallenge。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/// <summary>
/// Respond to an admin authentication challenge.
/// </summary>
/// <param name="userName">The name of the user.</param>
/// <param name="clientId">The client ID.</param>
/// <param name="mfaCode">The multi-factor authentication code.</param>
/// <param name="session">The current application session.</param>
/// <param name="clientId">The user pool ID.</param>
/// <returns>The result of the authentication response.</returns>
public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
    string userName,
    string clientId,
    string mfaCode,
    string session,
    string userPoolId)
{
```

```
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考AdminRespondToAuthChallenge中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";
```

```
   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
       request.AddChallengeResponses("USERNAME", userName);
       request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
       request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
       request.SetClientId(clientID);
       request.SetUserPoolId(userPoolID);
       request.SetSession(session);

       Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
               client.AdminRespondToAuthChallenge(request);

       if (outcome.IsSuccess()) {
           std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                     << std::endl;

           accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
       }
       else {
           std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考AdminRespondToAuthChallenge中的。

CLI

Amazon CLI

响应身份验证质询

可以通过多种方式响应不同的身份验证质询，具体取决于身份验证流程、用户池配置和用户设置。以下 `admin-respond-to-auth-challenge` 示例提供 diego@example.com 的 TOTP MFA 代码并完成登录。此用户池已启用设备记忆功能，因此身份验证结果还将返回新的设备密钥。

```
aws cognito-idp admin-respond-to-auth-challenge \
    --user-pool-id us-west-2_EXAMPLE \
    --client-id 1example23456789 \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-
responses USERNAME=diego@example.com,SOFTWARE_TOKEN_MFA_CODE=000000 \
    --session AYABeExample...
```

输出：

```
{
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-ExAmPlE1"
        }
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Admin authentication flow。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AdminRespondToAuthChallenge中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   // Respond to an authentication challenge.
   public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
           String userName, String clientId, String mfaCode, String session) {
       System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
       Map<String, String> challengeResponses = new HashMap<>();

       challengeResponses.put("USERNAME", userName);
       challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

       AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
               .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
               .clientId(clientId)
               .challengeResponses(challengeResponses)
               .session(session)
               .build();

       AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
               .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
               + respondToAuthChallengeResult.authenticationResult());
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考AdminRespondToAuthChallenge中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const adminRespondToAuthChallenge = ({
  userPoolId,
  clientId,
  username,
  totp,
  session,
}) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AdminRespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: totp,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考AdminRespondToAuthChallenge中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}
```

- 有关 API 的详细信息，请参阅适用AdminRespondToAuthChallenge于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

通过提供关联的 MFA 应用程序生成的代码来响应 MFA 质询。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
 of
        a two-factor sign-in. When sign-in is successful, it returns an access
 token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
 initiate
                        authentication.
```

```
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result
```

- 有关 API 的详细信息，请参阅适用AdminRespondToAuthChallenge于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
    ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
    ///     authenticator.
    ///   - session: The authentication session to continue processing.
    func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
 userName: String,
                                     clientId: String, userPoolId: String,
 mfaCode: String,
                                     session: String) async {
        print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

        var challengeResponsesOb: [String: String] = [:]
        challengeResponsesOb["USERNAME"] = userName
        challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

        do {
            let output = try await cipClient.adminRespondToAuthChallenge(
                input: AdminRespondToAuthChallengeInput(
```

```
                    challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                    challengeResponses: challengeResponsesOb,
                    clientId: clientId,
                    session: session,
                    userPoolId: userPoolId
                )
            )

            guard let authenticationResult = output.authenticationResult else {
                print("*** Unable to get authentication result.")
                return
            }

            print("=====> Authentication result (JWTs are redacted):")
            print(authenticationResult)
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用AdminRespondToAuthChallenge于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AdminSetUserPassword`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AdminSetUserPassword`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

CLI

Amazon CLI

以管理员身份设置用户密码

以下 `admin-set-user-password` 示例永久设置 diego@example.com 的密码。

```
aws cognito-idp admin-set-user-password \
    --user-pool-id us-west-2_EXAMPLE \
    --username diego@example.com \
    --password MyExamplePassword1! \
    --permanent
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Passwords, password recovery, and password policies。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AdminSetUserPassword中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考AdminSetUserPassword中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`AssociateSoftwareToken`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `AssociateSoftwareToken`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;
```

```
        Console.WriteLine($"Use the following secret code to set up the
authenticator: {secretCode}");

        return tokenResponse.Session;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考AssociateSoftwareToken中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
example Google Authenticator."
```

```
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考AssociateSoftwareToken中的。

CLI

Amazon CLI

为 MFA 身份验证器应用程序生成私有密钥

以下 `associate-software-token` 示例为已注册并收到访问令牌的用户生成 TOTP 私有密钥。可手动将生成的私有密钥输入到身份验证器应用程序中，或者应用程序可以将私有密钥呈现为用户可扫描的二维码。

```
aws cognito-idp associate-software-token \
    --access-token eyJra456defEXAMPLE
```

输出：

```
{
```

```
    "SecretCode": "QWERTYUIOP123456EXAMPLE"
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 TOTP software token MFA。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考AssociateSoftwareToken中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
       AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
               .session(session)
               .build();

       AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
               .associateSoftwareToken(softwareTokenRequest);
       String secretCode = tokenResponse.secretCode();
       System.out.println("Enter this token into Google Authenticator");
       System.out.println(secretCode);
       return tokenResponse.session();
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考AssociateSoftwareToken中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const associateSoftwareToken = (session) => {
  const client = new CognitoIdentityProviderClient({});
  const command = new AssociateSoftwareTokenCommand({
    Session: session,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考AssociateSoftwareToken中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }
```

```
    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}
```

- 有关 API 的详细信息，请参阅适用AssociateSoftwareToken于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
```

```
        self.client_secret = client_secret


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- 有关 API 的详细信息，请参阅适用AssociateSoftwareToken于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
 authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
 \(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
 app's MFA.")
            return nil
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用AssociateSoftwareToken于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmDevice`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmDevice`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
```

```
        return response.UserConfirmationNecessary;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ConfirmDevice中的。

CLI

Amazon CLI

确认用户设备

以下 `confirm-device` 示例为当前用户添加新的记忆设备。

```
aws cognito-idp confirm-device \
   --access-token eyJra456defEXAMPLE \
   --device-key us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111 \
   --device-secret-verifier-
config PasswordVerifier=TXlWZXJpZmllclN0cmluZw,Salt=TXlTUlBTYWx0
```

输出：

```
{
     "UserConfirmationNecessary": false
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Working with user devices in your user pool。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ConfirmDevice中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const confirmDevice = ({ deviceKey, accessToken, passwordVerifier, salt }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmDeviceCommand({
    DeviceKey: deviceKey,
    AccessToken: accessToken,
    DeviceSecretVerifierConfig: {
      PasswordVerifier: passwordVerifier,
      Salt: salt,
    },
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ConfirmDevice中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
```

```
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
```

```
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm
```

- 有关 API 的详细信息，请参阅适用ConfirmDevice于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmForgotPassword`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmForgotPassword`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 使用 Lambda 函数自动迁移已知用户

CLI

Amazon CLI

确认忘记的密码

此示例确认用户名 diego@example.com 的已忘记密码。

命令:

```
aws cognito-idp confirm-forgot-password --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --password PASSWORD --confirmation-code CONF_CODE
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ConfirmForgotPassword中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ConfirmForgotPassword中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ConfirmSignUp`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ConfirmSignUp`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
        return false;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ConfirmSignUp中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);

        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ConfirmSignUp中的。

CLI

Amazon CLI

确认注册

此示例确认用户名 diego@example.com 的注册。

命令:

```
aws cognito-idp confirm-sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username=diego@example.com --confirmation-code CONF_CODE
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ConfirmSignUp中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ConfirmSignUp中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ConfirmSignUp中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}
```

- 有关 API 的详细信息，请参阅适用ConfirmSignUp于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
```

```
        :param client_id: The ID of a client application registered with the user
pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True
```

- 有关 API 的详细信息，请参阅适用ConfirmSignUp于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
 `false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
 String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
```

```
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用ConfirmSignUp于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateUserPool`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateUserPool`。

CLI

Amazon CLI

创建最低配置的用户池

此示例创建了一个 MyUserPool 使用默认值命名的用户池。没有必要的属性，也没有应用程序客户端。MFA 和高级安全功能已禁用。

命令:

```
aws cognito-idp create-user-pool --pool-name MyUserPool
```

输出：

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
```

```
                "Mutable": false
            },
            {
                "Name": "name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
```

```
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
```

```
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
```

```
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
```

```
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "updated_at",
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547833345.777,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {},
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
```

```
        "CreationDate": 1547833345.777,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
        "LambdaConfig": {}
    }
}
```

创建具有两个必要属性的用户池

此示例创建了一个用户池 MyUserPool。该池配置为接受电子邮件作为用户名属性。它还使用 Amazon Simple Email Service 将电子邮件源地址设置为经过验证的地址。

命令:

```
aws cognito-idp create-user-pool --pool-name MyUserPool --username-
attributes "email" --email-configuration=SourceArn="arn:aws:ses:us-
east-1:111111111111:identity/
jane@example.com",ReplyToEmailAddress="jane@example.com"
```

输出：

```
{
  "UserPool": {
      "SchemaAttributes": [
          {
              "Name": "sub",
              "StringAttributeConstraints": {
                  "MinLength": "1",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": true,
              "AttributeDataType": "String",
              "Mutable": false
          },
          {
              "Name": "name",
              "StringAttributeConstraints": {
                  "MinLength": "0",
                  "MaxLength": "2048"
              },
              "DeveloperOnlyAttribute": false,
              "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "given_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "family_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "middle_name",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "nickname",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "preferred_username",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "profile",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "picture",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "website",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
```

```
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "email",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "email_verified",
                "Mutable": true
            },
            {
                "Name": "gender",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "birthdate",
                "StringAttributeConstraints": {
                    "MinLength": "10",
                    "MaxLength": "10"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
```

```
                "Name": "zoneinfo",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "locale",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "phone_number",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "AttributeDataType": "Boolean",
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "Name": "phone_number_verified",
                "Mutable": true
            },
            {
                "Name": "address",
                "StringAttributeConstraints": {
                    "MinLength": "0",
                    "MaxLength": "2048"
```

```
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "String",
                "Mutable": true
            },
            {
                "Name": "updated_at",
                "NumberAttributeConstraints": {
                    "MinValue": "0"
                },
                "DeveloperOnlyAttribute": false,
                "Required": false,
                "AttributeDataType": "Number",
                "Mutable": true
            }
        ],
        "MfaConfiguration": "OFF",
        "Name": "MyUserPool",
        "LastModifiedDate": 1547837788.189,
        "AdminCreateUserConfig": {
            "UnusedAccountValidityDays": 7,
            "AllowAdminCreateUserOnly": false
        },
        "EmailConfiguration": {
            "ReplyToEmailAddress": "jane@example.com",
            "SourceArn": "arn:aws:ses:us-east-1:111111111111:identity/
jane@example.com"
        },
        "Policies": {
            "PasswordPolicy": {
                "RequireLowercase": true,
                "RequireSymbols": true,
                "RequireNumbers": true,
                "MinimumLength": 8,
                "RequireUppercase": true
            }
        },
        "UsernameAttributes": [
            "email"
        ],
        "CreationDate": 1547837788.189,
        "EstimatedNumberOfUsers": 0,
        "Id": "us-west-2_aaaaaaaaa",
```

```
      "LambdaConfig": {}
  }
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateUserPool中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPool {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolName>\s
```

```
                Where:
                    userPoolName - The name to give your user pool when it's
created.
                """;

        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolName = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        String id = createPool(cognitoClient, userPoolName);
        System.out.println("User pool ID: " + id);
        cognitoClient.close();
    }

    public static String createPool(CognitoIdentityProviderClient cognitoClient,
String userPoolName) {
        try {
            CreateUserPoolRequest request = CreateUserPoolRequest.builder()
                    .poolName(userPoolName)
                    .build();

            CreateUserPoolResponse response =
cognitoClient.createUserPool(request);
            return response.userPool().id();

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
        return "";
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateUserPool中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`CreateUserPoolClient`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `CreateUserPoolClient`。

CLI

Amazon CLI

创建用户池客户端

以下`create-user-pool-client`示例创建了一个新的用户池客户端，该客户端具有客户端密钥、显式读取和写入属性、使用用户名密码和 SRP 流程登录、使用三个用户名密码登录、对 OAuth 范围子集的访问权限 IdPs、 PinPoint 分析和扩展的身份验证会话有效期。

```
aws cognito-idp create-user-pool-client \
    --user-pool-id us-west-2_EXAMPLE \
    --client-name MyTestClient \
    --generate-secret \
    --refresh-token-validity 10 \
    --access-token-validity 60 \
    --id-token-validity 60 \
    --token-validity-units AccessToken=minutes,IdToken=minutes,RefreshToken=days
 \
    --read-attributes email phone_number email_verified phone_number_verified \
    --write-attributes email phone_number \
    --explicit-auth-
flows ALLOW_USER_PASSWORD_AUTH ALLOW_USER_SRP_AUTH ALLOW_REFRESH_TOKEN_AUTH \
    --supported-identity-providers Google Facebook MyOIDC \
    --callback-urls https://www.amazon.com https://example.com http://
localhost:8001 myapp://example \
    --allowed-o-auth-flows code implicit \
    --allowed-o-auth-scopes openid profile aws.cognito.signin.user.admin solar-
system-data/asteroids.add \
    --allowed-o-auth-flows-user-pool-client \
    --analytics-configuration ApplicationArn=arn:aws:mobiletargeting:us-
west-2:767671399759:apps/thisisanexamplepinpointapplicationid,UserDataShared=TRUE
 \
    --prevent-user-existence-errors ENABLED \
    --enable-token-revocation \
    --enable-propagate-additional-user-context-data \
```

```
    --auth-session-validity 4
```

输出：

```
{
    "UserPoolClient": {
        "UserPoolId": "us-west-2_EXAMPLE",
        "ClientName": "MyTestClient",
        "ClientId": "123abc456defEXAMPLE",
        "ClientSecret": "this1234is5678my91011example1213client1415secret",
        "LastModifiedDate": 1726788459.464,
        "CreationDate": 1726788459.464,
        "RefreshTokenValidity": 10,
        "AccessTokenValidity": 60,
        "IdTokenValidity": 60,
        "TokenValidityUnits": {
            "AccessToken": "minutes",
            "IdToken": "minutes",
            "RefreshToken": "days"
        },
        "ReadAttributes": [
            "email_verified",
            "phone_number_verified",
            "phone_number",
            "email"
        ],
        "WriteAttributes": [
            "phone_number",
            "email"
        ],
        "ExplicitAuthFlows": [
            "ALLOW_USER_PASSWORD_AUTH",
            "ALLOW_USER_SRP_AUTH",
            "ALLOW_REFRESH_TOKEN_AUTH"
        ],
        "SupportedIdentityProviders": [
            "Google",
            "MyOIDC",
            "Facebook"
        ],
        "CallbackURLs": [
            "https://example.com",
            "https://www.amazon.com",
```

```
            "myapp://example",
            "http://localhost:8001"
        ],
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "AllowedOAuthScopes": [
            "aws.cognito.signin.user.admin",
            "openid",
            "profile",
            "solar-system-data/asteroids.add"
        ],
        "AllowedOAuthFlowsUserPoolClient": true,
        "AnalyticsConfiguration": {
            "ApplicationArn": "arn:aws:mobiletargeting:us-
west-2:123456789012:apps/thisisanexamplepinpointapplicationid",
            "RoleArn": "arn:aws:iam::123456789012:role/aws-service-role/cognito-
idp.amazonaws.com/AWSServiceRoleForAmazonCognitoIdp",
            "UserDataShared": true
        },
        "PreventUserExistenceErrors": "ENABLED",
        "EnableTokenRevocation": true,
        "EnablePropagateAdditionalUserContextData": true,
        "AuthSessionValidity": 4
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Application-specific settings with app clients。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考CreateUserPoolClient中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientReques
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CreateUserPoolClientRespon

/**
 * A user pool client app is an application that authenticates with Amazon
 * Cognito user pools.
 * When you create a user pool, you can configure app clients that allow mobile
 * or web applications
 * to call API operations to authenticate users, manage user attributes and
 * profiles,
 * and implement sign-up and sign-in flows.
 *
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class CreateUserPoolClient {
    public static void main(String[] args) {
        final String usage = """

                Usage:
                    <clientName> <userPoolId>\s

                Where:
                    clientName - The name for the user pool client to create.
                    userPoolId - The ID for the user pool.
                """;

        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }
```

```
        String clientName = args[0];
        String userPoolId = args[1];
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        createPoolClient(cognitoClient, clientName, userPoolId);
        cognitoClient.close();
    }

    public static void createPoolClient(CognitoIdentityProviderClient
 cognitoClient, String clientName,
            String userPoolId) {
        try {
            CreateUserPoolClientRequest request =
 CreateUserPoolClientRequest.builder()
                    .clientName(clientName)
                    .userPoolId(userPoolId)
                    .build();

            CreateUserPoolClientResponse response =
 cognitoClient.createUserPoolClient(request);
            System.out.println("User pool " +
 response.userPoolClient().clientName() + " created. ID: "
                    + response.userPoolClient().clientId());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考CreateUserPoolClient中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

DeleteUser与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 DeleteUser。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
```

```
                        << std::endl;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考DeleteUser中的。

CLI

Amazon CLI

删除用户

此示例删除一个用户。

命令:

```
aws cognito-idp delete-user --access-token ACCESS_TOKEN
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考DeleteUser中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)
```

```
type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考DeleteUser中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
```

```
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考DeleteUser中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ForgotPassword与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ForgotPassword`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 使用 Lambda 函数自动迁移已知用户

CLI

Amazon CLI

强制密码更改

以下的 `forgot-password` 示例向 jane@example.com 发送一条消息，要求他们更改密码。

```
aws cognito-idp forgot-password --client-id 38fjsnc484p94kpqsnet7mpld0 --
username jane@example.com
```

输出：

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
```

```
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ForgotPassword中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
```

```
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ForgotPassword中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`InitiateAuth`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `InitiateAuth`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 向需要 MFA 的用户池注册用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
```

```
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
 string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考InitiateAuth中的。

CLI

Amazon CLI

让用户登录

以下 `initiate-auth` 示例让用户使用基本的用户名/密码流程登录，无需进行其它质询。

```
aws cognito-idp initiate-auth \
    --auth-flow USER_PASSWORD_AUTH \
    --client-id 1example23456789 \
```

```
    --analytics-metadata AnalyticsEndpointId=d70b2ba36a8c4dc5a04a0451aEXAMPLE \
    --auth-parameters USERNAME=testuser,PASSWORD=[Password] --user-context-
data EncodedData=mycontextdata --client-metadata MyTestKey=MyTestValue
```

输出：

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Authentication。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考InitiateAuth中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
```

```
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考InitiateAuth中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const initiateAuth = ({ username, password, clientId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new InitiateAuthCommand({
    AuthFlow: AuthFlowType.USER_PASSWORD_AUTH,
    AuthParameters: {
      USERNAME: username,
      PASSWORD: password,
    },
    ClientId: clientId,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考InitiateAuth中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

此示例演示了如何使用被跟踪设备开始身份验证。要完成登录，客户端必须正确响应安全远程密码（SRP）质询。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.
```

```
        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
 {response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
```

```
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- 有关 API 的详细信息，请参阅适用InitiateAuth于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`ListUserPools`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListUserPools`。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
 ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListUserPools中的。

CLI

Amazon CLI

列出用户池

以下`list-user-pools`示例列出了当前 CLI 凭证 Amazon 账户中的 3 个可用用户池。

```
aws cognito-idp list-user-pools \
```

```
    --max-results 3
```

输出：

```
{
    "NextToken": "[Pagination token]",
    "UserPools": [
        {
            "CreationDate": 1681502497.741,
            "Id": "us-west-2_EXAMPLE1",
            "LambdaConfig": {
                "CustomMessage": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreSignUp": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGeneration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                "PreTokenGenerationConfig": {
                    "LambdaArn": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction",
                    "LambdaVersion": "V1_0"
                }
            },
            "LastModifiedDate": 1681502497.741,
            "Name": "user pool 1"
        },
        {
            "CreationDate": 1686064178.717,
            "Id": "us-west-2_EXAMPLE2",
            "LambdaConfig": {
            },
            "LastModifiedDate": 1686064178.873,
            "Name": "user pool 2"
        },
        {
            "CreationDate": 1627681712.237,
            "Id": "us-west-2_EXAMPLE3",
            "LambdaConfig": {
                "UserMigration": "arn:aws:lambda:us-
east-1:123456789012:function:MyFunction"
            },
            "LastModifiedDate": 1678486942.479,
            "Name": "user pool 3"
```

```
        }
    ]
}
```

有关更多信息，请参阅 Amazon Cognito 开发人员指南中的 Amazon Cognito 用户池。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListUserPools中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
package main

import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

// main uses the AWS SDK for Go V2 to create an Amazon Simple Notification
 Service
// (Amazon SNS) client and list the topics in your account.
// This example uses the default settings specified in your shared credentials
// and config files.
func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
```

```
 fmt.Println("Couldn't load default configuration. Have you set up your AWS
account?")
 fmt.Println(err)
 return
}
cognitoClient := cognitoidentityprovider.NewFromConfig(sdkConfig)
fmt.Println("Let's list the user pools for your account.")
var pools []types.UserPoolDescriptionType
paginator := cognitoidentityprovider.NewListUserPoolsPaginator(
 cognitoClient, &cognitoidentityprovider.ListUserPoolsInput{MaxResults:
aws.Int32(10)})
for paginator.HasMorePages() {
 output, err := paginator.NextPage(ctx)
 if err != nil {
  log.Printf("Couldn't get user pools. Here's why: %v\n", err)
 } else {
  pools = append(pools, output.UserPools...)
 }
}
if len(pools) == 0 {
 fmt.Println("You don't have any user pools!")
} else {
 for _, pool := range pools {
  fmt.Printf("\t%v: %v\n", *pool.Name, *pool.Id)
 }
}
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考ListUserPools中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUserPoolsRequest;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUserPools {
    public static void main(String[] args) {
        CognitoIdentityProviderClient cognitoClient =
 CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUserPools(cognitoClient);
        cognitoClient.close();
    }

    public static void listAllUserPools(CognitoIdentityProviderClient
 cognitoClient) {
        try {
            ListUserPoolsRequest request = ListUserPoolsRequest.builder()
                    .maxResults(10)
                    .build();

            ListUserPoolsResponse response =
 cognitoClient.listUserPools(request);
            response.userPools().forEach(userpool -> {
                System.out.println("User pool " + userpool.name() + ", User ID "
 + userpool.id());
            });
```

```
        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListUserPools中的。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client.list_user_pools().max_results(10).send().await?;
    let pools = response.user_pools();
    println!("User pools:");
    for pool in pools {
        println!("  ID:              {}", pool.id().unwrap_or_default());
        println!("  Name:            {}", pool.name().unwrap_or_default());
        println!("  Lambda Config:   {:?}", pool.lambda_config().unwrap());
        println!(
            "  Last modified:   {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!(
            "  Creation date:   {:?}",
            pool.creation_date().unwrap().to_chrono_utc()
        );
        println!();
    }
    println!("Next token: {}", response.next_token().unwrap_or_default());

    Ok(())
```

```
}
```

- 有关 API 的详细信息，请参阅适用ListUserPools于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ListUsers与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `ListUsers`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/// <summary>
/// Get a list of users for the Amazon Cognito user pool.
/// </summary>
/// <param name="userPoolId">The user pool ID.</param>
/// <returns>A list of users.</returns>
public async Task<List<UserType>> ListUsersAsync(string userPoolId)
{
    var request = new ListUsersRequest
    {
        UserPoolId = userPoolId
    };

    var users = new List<UserType>();
```

```
        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ListUsers中的。

CLI

Amazon CLI

示例 1：使用服务器端筛选条件列出用户

以下 `list-users` 示例列出了所请求的用户池中其电子邮件地址以 `testuser` 开头的 3 个用户。

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --filter email^=\"testuser\" \
    --max-items 3
```

输出：

```
{
    "PaginationToken": "efgh5678EXAMPLE",
    "Users": [
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "eaad0219-2117-439f-8d46-4db20e59268f"
                },
                {
                    "Name": "email",
                    "Value": "testuser@example.com"
                }
```

```
            ],
            "Enabled": true,
            "UserCreateDate": 1682955829.578,
            "UserLastModifiedDate": 1689030181.63,
            "UserStatus": "CONFIRMED",
            "Username": "testuser"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "3b994cfd-0b07-4581-be46-3c82f9a70c90"
                },
                {
                    "Name": "email",
                    "Value": "testuser2@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427979.201,
            "UserLastModifiedDate": 1684427979.201,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser2"
        },
        {
            "Attributes": [
                {
                    "Name": "sub",
                    "Value": "5929e0d1-4c34-42d1-9b79-a5ecacfe66f7"
                },
                {
                    "Name": "email",
                    "Value": "testuser3@example.com"
                }
            ],
            "Enabled": true,
            "UserCreateDate": 1684427823.641,
            "UserLastModifiedDate": 1684427823.641,
            "UserStatus": "UNCONFIRMED",
            "Username": "testuser3@example.com"
        }
    ]
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Managing and searching for users。

示例 2：使用客户端筛选条件列出用户

以下 `list-users` 示例列出了三个用户的属性，这些用户所具有的一个属性（在本例中为他们的电子邮件地址）包含电子邮件域“@example.com”。如果其它属性包含了此字符串，则也会显示这些属性。第二个用户没有与查询匹配的属性，因此会从显示的输出中排除，但不会从服务器响应中排除。

```
aws cognito-idp list-users \
    --user-pool-id us-west-2_EXAMPLE \
    --max-items 3
    --query Users\[\*\].Attributes\[\?Value\.contains\(\@\,\'@example.com\'\)\]
```

输出：

```
[
    [
        {
            "Name": "email",
            "Value": "admin@example.com"
        }
    ],
    [],
    [
        {
            "Name": "email",
            "Value": "operator@example.com"
        }
    ]
]
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Managing and searching for users。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ListUsers中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ListUsersResponse;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation topic:
 *
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 */
public class ListUsers {
    public static void main(String[] args) {

        final String usage = """

                Usage:
                    <userPoolId>\s

                Where:
                    userPoolId - The ID given to your user pool when it's
 created.
                """;
```

```
        if (args.length != 1) {
            System.out.println(usage);
            System.exit(1);
        }

        String userPoolId = args[0];
        CognitoIdentityProviderClient cognitoClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        listAllUsers(cognitoClient, userPoolId);
        listUsersFilter(cognitoClient, userPoolId);
        cognitoClient.close();
    }

    public static void listAllUsers(CognitoIdentityProviderClient cognitoClient,
String userPoolId) {
        try {
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User " + user.username() + " Status " +
user.userStatus() + " Created "
                        + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    // Shows how to list users by using a filter.
    public static void listUsersFilter(CognitoIdentityProviderClient
cognitoClient, String userPoolId) {

        try {
            String filter = "email = \"tblue@noserver.com\"";
            ListUsersRequest usersRequest = ListUsersRequest.builder()
                    .userPoolId(userPoolId)
```

```
                    .filter(filter)
                    .build();

            ListUsersResponse response = cognitoClient.listUsers(usersRequest);
            response.users().forEach(user -> {
                System.out.println("User with filter applied " + user.username()
 + " Status " + user.userStatus()
                        + " Created " + user.userCreateDate());
            });

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ListUsers中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const listUsers = ({ userPoolId }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ListUsersCommand({
    UserPoolId: userPoolId,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ListUsers中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun listAllUsers(userPoolId: String) {
    val request =
        ListUsersRequest {
            this.userPoolId = userPoolId
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use { cognitoClient ->
        val response = cognitoClient.listUsers(request)
        response.users?.forEach { user ->
            println("The user name is ${user.username}")
        }
    }
}
```

- 有关 API 的详细信息，请参阅适用ListUsers于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
 self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

- 有关 API 的详细信息，请参阅适用ListUsers于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

ResendConfirmationCode与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 ResendConfirmationCode。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };

        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);
```

```
        Console.WriteLine($"Method of delivery is
{response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考ResendConfirmationCode中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
```

```
                    << std::endl;
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考ResendConfirmationCode中的。

CLI

Amazon CLI

重新发送确认码

以下 `resend-confirmation-code` 示例向用户 `jane` 发送确认码。

```
aws cognito-idp resend-confirmation-code \
    --client-id 12a3b456c7de890f11g123hijk \
    --username jane
```

输出：

```
{
    "CodeDeliveryDetails": {
        "Destination": "j***@e***.com",
        "DeliveryMedium": "EMAIL",
        "AttributeName": "email"
    }
}
```

有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的注册并确认用户账户。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考ResendConfirmationCode中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
           String userName) {
       try {
           ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                   .clientId(clientId)
                   .username(userName)
                   .build();

           ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
           System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考ResendConfirmationCode中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const resendConfirmationCode = ({ clientId, username }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ResendConfirmationCodeCommand({
    ClientId: clientId,
    Username: username,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考ResendConfirmationCode中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
```

```
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}
```

- 有关 API 的详细信息，请参阅适用ResendConfirmationCode于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
```

```
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery
```

- 有关 API 的详细信息，请参阅适用ResendConfirmationCode于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
 clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
 else {
                print("*** Unable to get the delivery method for the resent
 code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
```

```
        } catch {
            print("*** Unable to resend the confirmation code to user
 \(userName).")
            return false
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用ResendConfirmationCode于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`RespondToAuthChallenge`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `RespondToAuthChallenge`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

CLI

Amazon CLI

示例 1：响应 NEW_PASSWORD_REQUIRED 质询

以下 `respond-to-auth-challenge` 示例响应了 initiate-auth 返回的 NEW_PASSWORD_REQUIRED 质询。它为用户 `jane@example.com` 设置密码。

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --challenge-name NEW_PASSWORD_REQUIRED \
    --challenge-responses USERNAME=jane@example.com,NEW_PASSWORD=[Password] \
    --session AYABeEv5HklEXAMPLE
```

输出：

```
{
```

```
    "ChallengeParameters": {},
    "AuthenticationResult": {
        "AccessToken": "ACCESS_TOKEN",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "REFRESH_TOKEN",
        "IdToken": "ID_TOKEN",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-wt2ha1Zd"
        }
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Authentication。

示例 2：响应 SELECT_MFA_TYPE 质询

以下 `respond-to-auth-challenge` 示例选择 TOTP MFA 作为当前用户的 MFA 选项。系统会提示用户选择 MFA 类型，接下来将提示用户输入其 MFA 代码。

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789
    --session AYABeEv5HklEXAMPLE
    --challenge-name SELECT_MFA_TYPE
    --challenge-responses USERNAME=testuser,ANSWER=SOFTWARE_TOKEN_MFA
```

输出：

```
{
    "ChallengeName": "SOFTWARE_TOKEN_MFA",
    "Session": "AYABeEv5HklEXAMPLE",
    "ChallengeParameters": {
        "FRIENDLY_DEVICE_NAME": "transparent"
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Adding MFA。

示例 3：响应 SOFTWARE_TOKEN_MFA 质询

以下 `respond-to-auth-challenge` 示例提供 TOTP MFA 代码并完成登录。

```
aws cognito-idp respond-to-auth-challenge \
    --client-id 1example23456789 \
    --session AYABeEv5HklEXAMPLE \
    --challenge-name SOFTWARE_TOKEN_MFA \
    --challenge-responses USERNAME=testuser,SOFTWARE_TOKEN_MFA_CODE=123456
```

输出：

```
{
    "AuthenticationResult": {
        "AccessToken": "eyJra456defEXAMPLE",
        "ExpiresIn": 3600,
        "TokenType": "Bearer",
        "RefreshToken": "eyJra123abcEXAMPLE",
        "IdToken": "eyJra789ghiEXAMPLE",
        "NewDeviceMetadata": {
            "DeviceKey": "us-west-2_a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "DeviceGroupKey": "-v7w9UcY6"
        }
    }
}
```

有关更多信息，请参阅《Amazon Cognito Developer Guide》中的 Adding MFA。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考RespondToAuthChallenge中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
```

```
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考RespondToAuthChallenge中的。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

登录跟踪的设备。要完成登录，客户端必须正确响应安全远程密码 (SRP) 质询。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
```

```
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
 an
                 access token for the user.
```

```
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )

            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
user_name})
            cr["USERNAME"] = user_name
```

```
            cr["DEVICE_KEY"] = device_key
            response_verifier =
self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

- 有关 API 的详细信息，请参阅适用RespondToAuthChallenge于 Python 的Amazon SDK (Boto3) API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

SignUp与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 SignUp。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
```

```
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考SignUp中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::SignUpRequest request;
        request.AddUserAttributes(
                Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                        "email").WithValue(email));
        request.SetUsername(userName);
        request.SetPassword(password);
        request.SetClientId(clientID);
        Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
                client.SignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "The signup request for " << userName << " was
successful."
                      << std::endl;
        }
        else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
            std::cout
```

```
                    << "The username already exists. Please enter a different
 username."
                    << std::endl;
            userExists = true;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考SignUp中的。

CLI

Amazon CLI

注册用户

此示例注册 jane@example.com。

命令:

```
aws cognito-idp sign-up --client-id 3n4b5urk1ft4fl3mg5e62d9ado --
username jane@example.com --password PASSWORD --user-attributes
 Name="email",Value="jane@example.com" Name="name",Value="Jane"
```

输出：

```
{
  "UserConfirmed": false,
  "UserSub": "e04d60a6-45dc-441c-a40b-e25a787d4862"
}
```

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考SignUp中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
```

```
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考SignUp中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    public static void signUp(CognitoIdentityProviderClient
 identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();
```

```
            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考SignUp中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考SignUp中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }

    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 有关 API 的详细信息，请参阅适用SignUp于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
 Cognito
        to send an email to the specified email address. The email contains a
 code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
 whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
```

```
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed
```

- 有关 API 的详细信息，请参阅适用SignUp于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
 userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )
```

```
            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }
```

- 如需了解 API 的详细信息，请参阅适用SignUp于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`UpdateUserPool`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `UpdateUserPool`。

操作示例是大型程序的代码摘录，必须在上下文中运行。您可以在以下代码示例中查看此操作的上下文：

- 使用 Lambda 函数自动确认已知用户
- 使用 Lambda 函数自动迁移已知用户
- 在完成 Amazon Cognito 用户身份验证后使用 Lambda 函数写入自定义活动数据

CLI

Amazon CLI

更新用户池

以下的 `update-user-pool` 示例使用每个可用配置选项的示例语法修改用户池。要更新用户池，必须指定所有先前配置的选项，否则这些选项将重置为默认值。

```
aws cognito-idp update-user-pool --user-pool-id us-west-2_EXAMPLE \
    --policies PasswordPolicy=
\{MinimumLength=6,RequireUppercase=true,RequireLowercase=true,RequireNumbers=true,Require
 \
    --deletion-protection ACTIVE \
    --lambda-config PreSignUp="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-presignup-
function",PreTokenGeneration="arn:aws:lambda:us-
west-2:123456789012:function:cognito-test-pretoken-function" \
    --auto-verified-attributes "phone_number" "email" \
    --verification-message-template \{\"SmsMessage\":\""Your code is
 {####}"\",\"EmailMessage\":\""Your code is {####}"\",\"EmailSubject\":\""Your
 verification code"\",\"EmailMessageByLink\":\""Click {##here##} to verify
 your email address."\",\"EmailSubjectByLink\":\""Your verification link"\",
\"DefaultEmailOption\":\"CONFIRM_WITH_LINK\"\} \
    --sms-authentication-message "Your code is {####}" \
    --user-attribute-update-settings
 AttributesRequireVerificationBeforeUpdate="email","phone_number" \
    --mfa-configuration "OPTIONAL" \
    --device-
configuration ChallengeRequiredOnNewDevice=true,DeviceOnlyRememberedOnUserPrompt=true
 \
    --email-configuration SourceArn="arn:aws:ses:us-
west-2:123456789012:identity/admin@example.com",ReplyToEmailAddress="amdin
+noreply@example.com",EmailSendingAccount=DEVELOPER,From="admin@amazon.com",Configuration
configuration-set" \
    --sms-configuration SnsCallerArn="arn:aws:iam::123456789012:role/service-
role/SNS-SMS-Role",ExternalId="12345",SnsRegion="us-west-2" \
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-tags "Function"="MyMobileGame","Developers"="Berlin" \
```

```
    --admin-create-user-config
 AllowAdminCreateUserOnly=false,InviteMessageTemplate=\{SMSMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailMessage=\""Welcome
 {username}. Your confirmation code is {####}"\",EmailSubject=\""Welcome to
 MyMobileGame"\"\} \
    --user-pool-add-ons AdvancedSecurityMode="AUDIT" \
    --account-recovery-setting RecoveryMechanisms=
\[\{Priority=1,Name="verified_email"\},
\{Priority=2,Name="verified_phone_number"\}\]
```

此命令不生成任何输出。

有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的更新用户池配置。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考UpdateUserPool中的。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}
```

```
// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
```

```
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}
```

- 有关 API 的详细信息，请参阅 适用于 Go 的 Amazon SDK API 参考UpdateUserPool中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
```

```
      },
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考UpdateUserPool中的。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

`VerifySoftwareToken`与 Amazon SDK 或 CLI 配合使用

以下代码示例演示如何使用 `VerifySoftwareToken`。

操作示例是大型程序的代码摘录，必须在上下文中运行。在以下代码示例中，您可以查看此操作的上下文：

- 向需要 MFA 的用户池注册用户

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
```

```
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }
```

- 有关 API 的详细信息，请参阅 适用于 .NET 的 Amazon SDK API 参考VerifySoftwareToken中的。

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

    Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
```

```
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
```

- 有关 API 的详细信息，请参阅 适用于 C++ 的 Amazon SDK API 参考VerifySoftwareToken中的。

CLI

Amazon CLI

确认 TOTP 身份验证器的注册

以下 `verify-software-token` 示例完成了当前用户的 TOTP 注册。

```
aws cognito-idp verify-software-token \
    --access-token eyJra456defEXAMPLE \
    --user-code 123456
```

输出：

```
{
    "Status": "SUCCESS"
}
```

有关更多信息，请参阅《Amazon Cognito 开发人员指南》中的向用户池添加 MFA。

- 有关 API 的详细信息，请参阅Amazon CLI 命令参考VerifySoftwareToken中的。

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }
```

- 有关 API 的详细信息，请参阅 Amazon SDK for Java 2.x API 参考VerifySoftwareToken中的。

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});

  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- 有关 API 的详细信息，请参阅 适用于 JavaScript 的 Amazon SDK API 参考VerifySoftwareToken中的。

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}
```

- 有关 API 的详细信息，请参阅适用VerifySoftwareToken于 K otlin 的Amazon SDK API 参考。

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.

        :param session: Session information returned from a previous call to
 initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response
```

- 有关 API 的详细信息，请参阅适用VerifySoftwareToken于 Python 的Amazon SDK (Boto3) API 参考。

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
import AWSCognitoIdentityProvider

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
 mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
```

```
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }
```

- 如需了解 API 的详细信息，请参阅适用VerifySoftwareToken于 S wift 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

亚马逊 Cognito 身份提供商使用的场景 Amazon SDKs

以下代码示例向您展示了如何使用在 Amazon Cognito 身份提供商中实现常见场景。 Amazon SDKs这些场景向您展示了如何通过在 Amazon Cognito 身份提供者中调用多个函数或结合其他 Amazon Web Services 服务来完成特定任务。每个场景都包含完整源代码的链接，您可以在其中找到有关如何设置和运行代码的说明。

场景以中等水平的经验为目标，可帮助您结合具体环境了解服务操作。

示例

- 使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 Amazon
- 使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 Amazon
- 使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 Amazon

- 使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 Amazon

使用软件开发工具包通过 Lambda 函数自动确认已知的亚马逊 Cognito 用户 Amazon

以下代码示例显示了如何使用 Lambda 函数自动确认已知的 Amazon Cognito 用户。

- 配置用户池以调用 `PreSignUp` 触发器的 Lambda 函数。
- 将用户注册到 Amazon Cognito
- Lambda 函数会扫描 DynamoDB 表并自动确认已知用户。
- 以新用户身份登录，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import (
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// AutoConfirm separates the steps of this scenario into individual functions so
 that
```

```
// they are simpler to read and understand.
type AutoConfirm struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewAutoConfirm constructs a new auto confirm runner.
func NewAutoConfirm(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) AutoConfirm {
 scenario := AutoConfirm{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddPreSignUpTrigger adds a Lambda handler as an invocation target for the
 PreSignUp trigger.
func (runner *AutoConfirm) AddPreSignUpTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the PreSignUp trigger from
 Cognito.\n" +
  "This trigger happens when a user signs up, and lets your function take action
 before the main Cognito\n" +
  "sign up processing occurs.\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PreSignUp, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the PreSignUp
 trigger.\n",
  functionArn, userPoolId)
}
```

```
// SignUpUser signs up a user from the known user table with a password you
 specify.
func (runner *AutoConfirm) SignUpUser(ctx context.Context, clientId string,
 usersTable string) (string, string) {
 log.Println("Let's sign up a user to your Cognito user pool. When the user's
 email matches an email in the\n" +
  "DynamoDB known users table, it is automatically verified and the user is
 confirmed.")

 knownUsers, err := runner.helper.GetKnownUsers(ctx, usersTable)
 if err != nil {
  panic(err)
 }
 userChoice := runner.questioner.AskChoice("Which user do you want to use?\n",
 knownUsers.UserNameList())
 user := knownUsers.Users[userChoice]

 var signedUp bool
 var userConfirmed bool
 password := runner.questioner.AskPassword("Enter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !signedUp {
  log.Printf("Signing up user '%v' with email '%v' to Cognito.\n", user.UserName,
 user.UserEmail)
  userConfirmed, err = runner.cognitoActor.SignUp(ctx, clientId, user.UserName,
 password, user.UserEmail)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("Enter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   signedUp = true
  }
 }
 log.Printf("User %v signed up, confirmed = %v.\n", user.UserName, userConfirmed)

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}
```

```
// SignInUser signs in a user.
func (runner *AutoConfirm) SignInUser(ctx context.Context, clientId string,
 userName string, password string) string {
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 log.Printf("Let's sign in as %v...\n", userName)
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Printf("Successfully signed in. Your access token starts with: %v...\n",
 (*authResult.AccessToken)[:10])
 log.Println(strings.Repeat("-", 88))
 return *authResult.AccessToken
}

// Run runs the scenario.
func (runner *AutoConfirm) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])

 runner.AddPreSignUpTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["AutoConfirmFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers, actions.PreSignUp)
 userName, password := runner.SignUpUser(ctx, stackOutputs["UserPoolClientId"],
 stackOutputs["TableName"])
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["AutoConfirmFunction"])
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
```

```
  runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password))

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函数处理 PreSignUp 触发器。

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
 if err != nil {
  panic(err)
 }
```

```
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PreSignUp event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be confirmed and verified.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsPreSignup) (events.CognitoEventUserPoolsPreSignup,
 error) {
 log.Printf("Received presignup from %v for user '%v'", event.TriggerSource,
 event.UserName)
 if event.TriggerSource != "PreSignUp_SignUp" {
  // Other trigger sources, such as PreSignUp_AdminInitiateAuth, ignore the
 response from this handler.
  return event, nil
 }
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserEmail: event.Request.UserAttributes["email"],
 }
 log.Printf("Looking up email %v in table %v.\n", user.UserEmail, tableName)
 output, err := h.dynamoClient.GetItem(ctx, &dynamodb.GetItemInput{
  Key:       user.GetKey(),
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Error looking up email %v.\n", user.UserEmail)
  return event, err
 }
 if output.Item == nil {
  log.Printf("Email %v not found. Email verification is required.\n",
 user.UserEmail)
  return event, err
 }

 err = attributevalue.UnmarshalMap(output.Item, &user)
 if err != nil {
  log.Printf("Couldn't unmarshal DynamoDB item. Here's why: %v\n", err)
  return event, err
 }
```

```
 if user.UserName != event.UserName {
  log.Printf("UserEmail %v found, but stored UserName '%v' does not match
 supplied UserName '%v'. Verification is required.\n",
   user.UserEmail, user.UserName, event.UserName)
 } else {
  log.Printf("UserEmail %v found with matching UserName %v. User is confirmed.
\n", user.UserEmail, user.UserName)
  event.Response.AutoConfirmUser = true
  event.Response.AutoVerifyEmail = true
 }

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

创建一个执行常见任务的结构。

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
```

```
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
```

```
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}
```

```
// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
```

```
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int

const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
```

```
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
```

```
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}
```

```
// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
```

```
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
```

```
}
```

创建一个封装 DynamoDB 操作的结构。

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}
```

```
// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
```

```
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
```

```
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
```

```
  events = output.Events
 }
 return events, err
}
```

创建一个封装动作的结构。 Amazon CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
import (
 "context"
 "log"
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()
```

```
 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
   }
   log.Println("Deleted user.")
  }
  triggerList := make([]actions.TriggerInfo, len(resources.triggers))
  for i := 0; i < len(resources.triggers); i++ {
   triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
 HandlerArn: nil}
  }
  err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
 triggerList...)
  if err != nil {
   log.Println("Couldn't update Cognito triggers during cleanup.")
   panic(err)
  }
  log.Println("Removed Cognito triggers from user pool.")
 } else {
  log.Println("Be sure to remove resources when you're done with them to avoid
 unexpected charges!")
 }
}
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API 参考》中的以下主题。
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

配置交互式“场景”运行。 JavaScript (v3) 示例共享一个场景运行器，以简化复杂的示例。完整的源代码已打开 GitHub。

```
import { AutoConfirm } from "./scenario-auto-confirm.js";

/**
 * The context is passed to every scenario. Scenario steps
 * will modify the context.
 */
const context = {
  errors: [],
  users: [
    {
      UserName: "test_user_1",
      UserEmail: "test_email_1@example.com",
    },
    {
      UserName: "test_user_2",
      UserEmail: "test_email_2@example.com",
    },
    {
      UserName: "test_user_3",
      UserEmail: "test_email_3@example.com",
    },
  ],
};

/**
 * Three Scenarios are created for the workflow. A Scenario is an orchestration
 class
 * that simplifies running a series of steps.
 */
```

```
export const scenarios = {
  // Demonstrate automatically confirming known users in a database.
  "auto-confirm": AutoConfirm(context),
};

// Call function if run directly
import { fileURLToPath } from "node:url";
import { parseScenarioArgs } from "@aws-doc-sdk-examples/lib/scenario/index.js";

if (process.argv[1] === fileURLToPath(import.meta.url)) {
  parseScenarioArgs(scenarios, {
    name: "Cognito user pools and triggers",
    description:
      "Demonstrate how to use the AWS SDKs to customize Amazon Cognito
 authentication behavior.",
  });
}
```

此场景演示了如何自动确认已知用户。其编排了示例步骤。

```
import { wait } from "@aws-doc-sdk-examples/lib/utils/util-timers.js";
import {
  Scenario,
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";

import {
  getStackOutputs,
  logCleanUpReminder,
  promptForStackName,
  promptForStackRegion,
  skipWhenErrors,
} from "./steps-common.js";
import { populateTable } from "./actions/dynamodb-actions.js";
import {
  addPreSignUpHandler,
  deleteUser,
  getUser,
  signIn,
  signUpUser,
```

```
} from "./actions/cognito-actions.js";
import {
  getLatestLogStreamForLambda,
  getLogEvents,
} from "./actions/cloudwatch-logs-actions.js";

/**
 * @typedef {{
 *   errors: Error[],
 *   password: string,
 *   users: { UserName: string, UserEmail: string }[],
 *   selectedUser?: string,
 *   stackName?: string,
 *   stackRegion?: string,
 *   token?: string,
 *   confirmDeleteSignedInUser?: boolean,
 *   TableName?: string,
 *   UserPoolClientId?: string,
 *   UserPoolId?: string,
 *   UserPoolArn?: string,
 *   AutoConfirmHandlerArn?: string,
 *   AutoConfirmHandlerName?: string
 * }} State
 */

const greeting = new ScenarioOutput(
  "greeting",
  (/** @type {State} */ state) => `This demo will populate some users into the \
database created as part of the "${state.stackName}" stack. \
Then the AutoConfirmHandler will be linked to the PreSignUp \
trigger from Cognito. Finally, you will choose a user to sign up.`,
  { skipWhen: skipWhenErrors },
);

const logPopulatingUsers = new ScenarioOutput(
  "logPopulatingUsers",
  "Populating the DynamoDB table with some users.",
  { skipWhenErrors: skipWhenErrors },
);

const logPopulatingUsersComplete = new ScenarioOutput(
  "logPopulatingUsersComplete",
  "Done populating users.",
  { skipWhen: skipWhenErrors },
```

```
);

const populateUsers = new ScenarioAction(
  "populateUsers",
  async (/** @type {State} */ state) => {
    const [_, err] = await populateTable({
      region: state.stackRegion,
      tableName: state.TableName,
      items: state.users,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTrigger = new ScenarioOutput(
  "logSetupSignUpTrigger",
  "Setting up the PreSignUp trigger for the Cognito User Pool.",
  { skipWhen: skipWhenErrors },
);

const setupSignUpTrigger = new ScenarioAction(
  "setupSignUpTrigger",
  async (/** @type {State} */ state) => {
    const [_, err] = await addPreSignUpHandler({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      handlerArn: state.AutoConfirmHandlerArn,
    });
    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const logSetupSignUpTriggerComplete = new ScenarioOutput(
  "logSetupSignUpTriggerComplete",
```

```
  (
    /** @type {State} */ state,
  ) => `The lambda function "${state.AutoConfirmHandlerName}" \
has been configured as the PreSignUp trigger handler for the user pool
 "${state.UserPoolId}".`,
  { skipWhen: skipWhenErrors },
);

const selectUser = new ScenarioInput(
  "selectedUser",
  "Select a user to sign up.",
  {
    type: "select",
    choices: (/** @type {State} */ state) => state.users.map((u) => u.UserName),
    skipWhen: skipWhenErrors,
    default: (/** @type {State} */ state) => state.users[0].UserName,
  },
);

const checkIfUserAlreadyExists = new ScenarioAction(
  "checkIfUserAlreadyExists",
  async (/** @type {State} */ state) => {
    const [user, err] = await getUser({
      region: state.stackRegion,
      userPoolId: state.UserPoolId,
      username: state.selectedUser,
    });

    if (err?.name === "UserNotFoundException") {
      // Do nothing. We're not expecting the user to exist before
      // sign up is complete.
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    if (user) {
      state.errors.push(
        new Error(
          `The user "${state.selectedUser}" already exists in the user pool
 "${state.UserPoolId}".`,
```

```
        ),
      );
    }
  },
  {
    skipWhen: skipWhenErrors,
  },
);

const createPassword = new ScenarioInput(
  "password",
  "Enter a password that has at least eight characters, uppercase, lowercase,
 numbers and symbols.",
  { type: "password", skipWhen: skipWhenErrors, default: "Abcd1234!" },
);

const logSignUpExistingUser = new ScenarioOutput(
  "logSignUpExistingUser",
  (/** @type {State} */ state) => `Signing up user "${state.selectedUser}".`,
  { skipWhen: skipWhenErrors },
);

const signUpExistingUser = new ScenarioAction(
  "signUpExistingUser",
  async (/** @type {State} */ state) => {
    const signUp = (password) =>
      signUpUser({
        region: state.stackRegion,
        userPoolClientId: state.UserPoolClientId,
        username: state.selectedUser,
        email: state.users.find((u) => u.UserName === state.selectedUser)
          .UserEmail,
        password,
      });

    let [_, err] = await signUp(state.password);

    while (err?.name === "InvalidPasswordException") {
      console.warn("The password you entered was invalid.");
      await createPassword.handle(state);
      [_, err] = await signUp(state.password);
    }

    if (err) {
```

```
      state.errors.push(err);
    }
  },
  { skipWhen: skipWhenErrors },
);

const logSignUpExistingUserComplete = new ScenarioOutput(
  "logSignUpExistingUserComplete",
  (/** @type {State} */ state) =>
    `"${state.selectedUser} was signed up successfully.`,
  { skipWhen: skipWhenErrors },
);

const logLambdaLogs = new ScenarioAction(
  "logLambdaLogs",
  async (/** @type {State} */ state) => {
    console.log(
      "Waiting a few seconds to let Lambda write to CloudWatch Logs...\n",
    );
    await wait(10);

    const [logStream, logStreamErr] = await getLatestLogStreamForLambda({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
    });
    if (logStreamErr) {
      state.errors.push(logStreamErr);
      return;
    }

    console.log(
      `Getting some recent events from log stream "${logStream.logStreamName}"`,
    );
    const [logEvents, logEventsErr] = await getLogEvents({
      functionName: state.AutoConfirmHandlerName,
      region: state.stackRegion,
      eventCount: 10,
      logStreamName: logStream.logStreamName,
    });
    if (logEventsErr) {
      state.errors.push(logEventsErr);
      return;
    }
```

```
    console.log(logEvents.map((ev) => `\t${ev.message}`).join(""));
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUser = new ScenarioOutput(
  "logSignInUser",
  (/** @type {State} */ state) => `Let's sign in as ${state.selectedUser}`,
  { skipWhen: skipWhenErrors },
);

const signInUser = new ScenarioAction(
  "signInUser",
  async (/** @type {State} */ state) => {
    const [response, err] = await signIn({
      region: state.stackRegion,
      clientId: state.UserPoolClientId,
      username: state.selectedUser,
      password: state.password,
    });

    if (err?.name === "PasswordResetRequiredException") {
      state.errors.push(new Error("Please reset your password."));
      return;
    }

    if (err) {
      state.errors.push(err);
      return;
    }

    state.token = response?.AuthenticationResult?.AccessToken;
  },
  { skipWhen: skipWhenErrors },
);

const logSignInUserComplete = new ScenarioOutput(
  "logSignInUserComplete",
  (/** @type {State} */ state) =>
    `Successfully signed in. Your access token starts with:
 ${state.token.slice(0, 11)}`,
  { skipWhen: skipWhenErrors },
);
```

```
const confirmDeleteSignedInUser = new ScenarioInput(
  "confirmDeleteSignedInUser",
  "Do you want to delete the currently signed in user?",
  { type: "confirm", skipWhen: skipWhenErrors },
);

const deleteSignedInUser = new ScenarioAction(
  "deleteSignedInUser",
  async (/** @type {State} */ state) => {
    const [_, err] = await deleteUser({
      region: state.stackRegion,
      accessToken: state.token,
    });

    if (err) {
      state.errors.push(err);
    }
  },
  {
    skipWhen: (/** @type {State} */ state) =>
      skipWhenErrors(state) || !state.confirmDeleteSignedInUser,
  },
);

const logErrors = new ScenarioOutput(
  "logErrors",
  (/** @type {State}*/ state) => {
    const errorList = state.errors
      .map((err) => ` - ${err.name}: ${err.message}`)
      .join("\n");
    return `Scenario errors found:\n${errorList}`;
  },
  {
    // Don't log errors when there aren't any!
    skipWhen: (/** @type {State} */ state) => state.errors.length === 0,
  },
);

export const AutoConfirm = (context) =>
  new Scenario(
    "AutoConfirm",
    [
      promptForStackName,
      promptForStackRegion,
```

```
      getStackOutputs,
      greeting,
      logPopulatingUsers,
      populateUsers,
      logPopulatingUsersComplete,
      logSetupSignUpTrigger,
      setupSignUpTrigger,
      logSetupSignUpTriggerComplete,
      selectUser,
      checkIfUserAlreadyExists,
      createPassword,
      logSignUpExistingUser,
      signUpExistingUser,
      logSignUpExistingUserComplete,
      logLambdaLogs,
      logSignInUser,
      signInUser,
      logSignInUserComplete,
      confirmDeleteSignedInUser,
      deleteSignedInUser,
      logCleanUpReminder,
      logErrors,
    ],
    context,
  );
```

这些步骤与其他场景共享。

```
import {
  ScenarioAction,
  ScenarioInput,
  ScenarioOutput,
} from "@aws-doc-sdk-examples/lib/scenario/scenario.js";
import { getCfnOutputs } from "@aws-doc-sdk-examples/lib/sdk/cfn-outputs.js";

export const skipWhenErrors = (state) => state.errors.length > 0;

export const getStackOutputs = new ScenarioAction(
  "getStackOutputs",
  async (state) => {
    if (!state.stackName || !state.stackRegion) {
      state.errors.push(
```

```
        new Error(
          "No stack name or region provided. The stack name and \
region are required to fetch CFN outputs relevant to this example.",
        ),
      );
      return;
    }

    const outputs = await getCfnOutputs(state.stackName, state.stackRegion);
    Object.assign(state, outputs);
  },
);

export const promptForStackName = new ScenarioInput(
  "stackName",
  "Enter the name of the stack you deployed earlier.",
  { type: "input", default: "PoolsAndTriggersStack" },
);

export const promptForStackRegion = new ScenarioInput(
  "stackRegion",
  "Enter the region of the stack you deployed earlier.",
  { type: "input", default: "us-east-1" },
);

export const logCleanUpReminder = new ScenarioOutput(
  "logCleanUpReminder",
  "All done. Remember to run 'cdk destroy' to teardown the stack.",
  { skipWhen: skipWhenErrors },
);
```

具有 Lambda 函数的 `PreSignUp` 触发器的处理程序。

```
import type { PreSignUpTriggerEvent, Handler } from "aws-lambda";
import type { UserRepository } from "./user-repository";
import { DynamoDBUserRepository } from "./user-repository";

export class PreSignUpHandler {
  private userRepository: UserRepository;

  constructor(userRepository: UserRepository) {
    this.userRepository = userRepository;
```

```
  }

  private isPreSignUpTriggerSource(event: PreSignUpTriggerEvent): boolean {
    return event.triggerSource === "PreSignUp_SignUp";
  }

  private getEventUserEmail(event: PreSignUpTriggerEvent): string {
    return event.request.userAttributes.email;
  }

  async handlePreSignUpTriggerEvent(
    event: PreSignUpTriggerEvent,
  ): Promise<PreSignUpTriggerEvent> {
    console.log(
      `Received presignup from ${event.triggerSource} for user
'${event.userName}'`,
    );

    if (!this.isPreSignUpTriggerSource(event)) {
      return event;
    }

    const eventEmail = this.getEventUserEmail(event);
    console.log(`Looking up email ${eventEmail}.`);
    const storedUserInfo =
      await this.userRepository.getUserInfoByEmail(eventEmail);

    if (!storedUserInfo) {
      console.log(
        `Email ${eventEmail} not found. Email verification is required.`,
      );
      return event;
    }

    if (storedUserInfo.UserName !== event.userName) {
      console.log(
        `UserEmail ${eventEmail} found, but stored UserName
'${storedUserInfo.UserName}' does not match supplied UserName
'${event.userName}'. Verification is required.`,
      );
    } else {
      console.log(
        `UserEmail ${eventEmail} found with matching UserName
${storedUserInfo.UserName}. User is confirmed.`,
```

```
      );
      event.response.autoConfirmUser = true;
      event.response.autoVerifyEmail = true;
    }
    return event;
  }
}

const createPreSignUpHandler = (): PreSignUpHandler => {
  const tableName = process.env.TABLE_NAME;
  if (!tableName) {
    throw new Error("TABLE_NAME environment variable is not set");
  }

  const userRepository = new DynamoDBUserRepository(tableName);
  return new PreSignUpHandler(userRepository);
};

export const handler: Handler = async (event: PreSignUpTriggerEvent) => {
  const preSignUpHandler = createPreSignUpHandler();
  return preSignUpHandler.handlePreSignUpTriggerEvent(event);
};
```

CloudWatch 日志操作模块。

```
import {
  CloudWatchLogsClient,
  GetLogEventsCommand,
  OrderBy,
  paginateDescribeLogStreams,
} from "@aws-sdk/client-cloudwatch-logs";

/**
 * Get the latest log stream for a Lambda function.
 * @param {{ functionName: string, region: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").LogStream | null,
 unknown]>}
 */
export const getLatestLogStreamForLambda = async ({ functionName, region }) => {
  try {
    const logGroupName = `/aws/lambda/${functionName}`;
```

```
    const cwlClient = new CloudWatchLogsClient({ region });
    const paginator = paginateDescribeLogStreams(
      { client: cwlClient },
      {
        descending: true,
        limit: 1,
        orderBy: OrderBy.LastEventTime,
        logGroupName,
      },
    );

    for await (const page of paginator) {
      return [page.logStreams[0], null];
    }
  } catch (err) {
    return [null, err];
  }
};

/**
 * Get the log events for a Lambda function's log stream.
 * @param {{
 *   functionName: string,
 *   logStreamName: string,
 *   eventCount: number,
 *   region: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cloudwatch-logs").OutputLogEvent[]
 | null, unknown]>}
 */
export const getLogEvents = async ({
  functionName,
  logStreamName,
  eventCount,
  region,
}) => {
  try {
    const cwlClient = new CloudWatchLogsClient({ region });
    const logGroupName = `/aws/lambda/${functionName}`;
    const response = await cwlClient.send(
      new GetLogEventsCommand({
        logStreamName: logStreamName,
        limit: eventCount,
        logGroupName: logGroupName,
```

```
      }),
    );

    return [response.events, null];
  } catch (err) {
    return [null, err];
  }
};
```

Amazon Cognito 操作的模块。

```
import {
  AdminGetUserCommand,
  CognitoIdentityProviderClient,
  DeleteUserCommand,
  InitiateAuthCommand,
  SignUpCommand,
  UpdateUserPoolCommand,
} from "@aws-sdk/client-cognito-identity-provider";

/**
 * Connect a Lambda function to the PreSignUp trigger for a Cognito user pool
 * @param {{ region: string, userPoolId: string, handlerArn: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").UpdateUserPoolCommandOutput | null, unknown]>}
 */
export const addPreSignUpHandler = async ({
  region,
  userPoolId,
  handlerArn,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const command = new UpdateUserPoolCommand({
      UserPoolId: userPoolId,
      LambdaConfig: {
        PreSignUp: handlerArn,
      },
```

```
    });

    const response = await cognitoClient.send(command);
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Attempt to register a user to a user pool with a given username and password.
 * @param {{
 *   region: string,
 *   userPoolClientId: string,
 *   username: string,
 *   email: string,
 *   password: string
 * }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").SignUpCommandOutput | null, unknown]>}
 */
export const signUpUser = async ({
  region,
  userPoolClientId,
  username,
  email,
  password,
}) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({
      region,
    });

    const response = await cognitoClient.send(
      new SignUpCommand({
        ClientId: userPoolClientId,
        Username: username,
        Password: password,
        UserAttributes: [{ Name: "email", Value: email }],
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
```

```
  }
};

/**
 * Sign in a user to Amazon Cognito using a username and password authentication
 flow.
 * @param {{ region: string, clientId: string, username: string, password:
 string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").InitiateAuthCommandOutput | null, unknown]>}
 */
export const signIn = async ({ region, clientId, username, password }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new InitiateAuthCommand({
        AuthFlow: "USER_PASSWORD_AUTH",
        ClientId: clientId,
        AuthParameters: { USERNAME: username, PASSWORD: password },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};

/**
 * Retrieve an existing user from a user pool.
 * @param {{ region: string, userPoolId: string, username: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").AdminGetUserCommandOutput | null, unknown]>}
 */
export const getUser = async ({ region, userPoolId, username }) => {
  try {
    const cognitoClient = new CognitoIdentityProviderClient({ region });
    const response = await cognitoClient.send(
      new AdminGetUserCommand({
        UserPoolId: userPoolId,
        Username: username,
      }),
    );
    return [response, null];
  } catch (err) {
```

```
    return [null, err];
  }
};

/**
 * Delete the signed-in user. Useful for allowing a user to delete their
 * own profile.
 * @param {{ region: string, accessToken: string }} config
 * @returns {Promise<[import("@aws-sdk/client-cognito-identity-
provider").DeleteUserCommandOutput | null, unknown]>}
 */
export const deleteUser = async ({ region, accessToken }) => {
  try {
    const client = new CognitoIdentityProviderClient({ region });
    const response = await client.send(
      new DeleteUserCommand({ AccessToken: accessToken }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

DynamoDB 操作的模块。

```
import { DynamoDBClient } from "@aws-sdk/client-dynamodb";
import {
  BatchWriteCommand,
  DynamoDBDocumentClient,
} from "@aws-sdk/lib-dynamodb";

/**
 * Populate a DynamoDB table with provide items.
 * @param {{ region: string, tableName: string, items: Record<string,
 unknown>[] }} config
 * @returns {Promise<[import("@aws-sdk/lib-dynamodb").BatchWriteCommandOutput |
 null, unknown]>}
 */
export const populateTable = async ({ region, tableName, items }) => {
  try {
    const ddbClient = new DynamoDBClient({ region });
```

```
    const docClient = DynamoDBDocumentClient.from(ddbClient);
    const response = await docClient.send(
      new BatchWriteCommand({
        RequestItems: {
          [tableName]: items.map((item) => ({
            PutRequest: {
              Item: item,
            },
          })),
        },
      }),
    );
    return [response, null];
  } catch (err) {
    return [null, err];
  }
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的以下主题。
  - DeleteUser
  - InitiateAuth
  - SignUp
  - UpdateUserPool

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包使用 Lambda 函数自动迁移已知的亚马逊 Cognito 用户 Amazon

以下代码示例显示了如何使用 Lambda 函数自动迁移已知的 Amazon Cognito 用户。

- 配置用户池以调用 `MigrateUser` 触发器的 Lambda 函数。
- 使用不在用户池中的用户名和电子邮件地址登录 Amazon Cognito。
- Lambda 函数会扫描 DynamoDB 表并自动将已知用户迁移到该用户池。
- 执行“忘记密码”流程可重置已迁移用户的密码。
- 以新用户身份登录，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import (
 "context"
 "errors"
 "fmt"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// MigrateUser separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type MigrateUser struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewMigrateUser constructs a new migrate user runner.
func NewMigrateUser(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) MigrateUser {
 scenario := MigrateUser{
  helper:       helper,
  questioner:   questioner,
```

```
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddMigrateUserTrigger adds a Lambda handler as an invocation target for the
 MigrateUser trigger.
func (runner *MigrateUser) AddMigrateUserTrigger(ctx context.Context, userPoolId
 string, functionArn string) {
 log.Printf("Let's add a Lambda function to handle the MigrateUser trigger from
 Cognito.\n" +
  "This trigger happens when an unknown user signs in, and lets your function
 take action before Cognito\n" +
  "rejects the user.\n\n")
 err := runner.cognitoActor.UpdateTriggers(
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.UserMigration, HandlerArn:
 aws.String(functionArn)})
 if err != nil {
  panic(err)
 }
 log.Printf("Lambda function %v added to user pool %v to handle the MigrateUser
 trigger.\n",
  functionArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser adds a new user to the known users table and signs that user in to
 Amazon Cognito.
func (runner *MigrateUser) SignInUser(ctx context.Context, usersTable string,
 clientId string) (bool, actions.User) {
 log.Println("Let's sign in a user to your Cognito user pool. When the username
 and email matches an entry in the\n" +
  "DynamoDB known users table, the email is automatically verified and the user
 is migrated to the Cognito user pool.")

 user := actions.User{}
 user.UserName = runner.questioner.Ask("\nEnter a username:")
 user.UserEmail = runner.questioner.Ask("\nEnter an email that you own. This
 email will be used to confirm user migration\n" +
```

```
  "during this example:")

 runner.helper.AddKnownUser(ctx, usersTable, user)

 var err error
 var resetRequired *types.PasswordResetRequiredException
 var authResult *types.AuthenticationResultType
 signedIn := false
 for !signedIn && resetRequired == nil {
  log.Printf("Signing in to Cognito as user '%v'. The expected result is a
 PasswordResetRequiredException.\n\n", user.UserName)
  authResult, err = runner.cognitoActor.SignIn(ctx, clientId, user.UserName, "_")
  if err != nil {
   if errors.As(err, &resetRequired) {
    log.Printf("\nUser '%v' is not in the Cognito user pool but was found in the
 DynamoDB known users table.\n"+
     "User migration is started and a password reset is required.",
 user.UserName)
   } else {
    panic(err)
   }
  } else {
   log.Printf("User '%v' successfully signed in. This is unexpected and probably
 means you have not\n"+
    "cleaned up a previous run of this scenario, so the user exist in the Cognito
 user pool.\n"+
    "You can continue this example and select to clean up resources, or manually
 remove\n"+
    "the user from your user pool and try again.", user.UserName)
   runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
   signedIn = true
  }
 }

 log.Println(strings.Repeat("-", 88))
 return resetRequired != nil, user
}

// ResetPassword starts a password recovery flow.
func (runner *MigrateUser) ResetPassword(ctx context.Context, clientId string,
 user actions.User) {
 wantCode := runner.questioner.AskBool(fmt.Sprintf("In order to migrate the user
 to Cognito, you must be able to receive a confirmation\n"+
```

```
 "code by email at %v. Do you want to send a code (y/n)?", user.UserEmail), "y")
if !wantCode {
 log.Println("To complete this example and successfully migrate a user to
Cognito, you must enter an email\n" +
  "you own that can receive a confirmation code.")
 return
}
codeDelivery, err := runner.cognitoActor.ForgotPassword(ctx, clientId,
user.UserName)
if err != nil {
 panic(err)
}
log.Printf("\nA confirmation code has been sent to %v.",
*codeDelivery.Destination)
code := runner.questioner.Ask("Check your email and enter it here:")

confirmed := false
password := runner.questioner.AskPassword("\nEnter a password that has at least
eight characters, uppercase, lowercase, numbers and symbols.\n"+
 "(the password will not display as you type):", 8)
for !confirmed {
 log.Printf("\nConfirming password reset for user '%v'.\n", user.UserName)
 err = runner.cognitoActor.ConfirmForgotPassword(ctx, clientId, code,
user.UserName, password)
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   password = runner.questioner.AskPassword("\nEnter another password:", 8)
  } else {
   panic(err)
  }
 } else {
  confirmed = true
 }
}
log.Printf("User '%v' successfully confirmed and migrated.\n", user.UserName)
log.Println("Signing in with your username and password...")
authResult, err := runner.cognitoActor.SignIn(ctx, clientId, user.UserName,
password)
if err != nil {
 panic(err)
}
log.Printf("Successfully signed in. Your access token starts with: %v...\n",
(*authResult.AccessToken)[:10])
```

```
 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)

 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *MigrateUser) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]

 runner.AddMigrateUserTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["MigrateUserFunctionArn"])
 runner.resources.triggers = append(runner.resources.triggers,
 actions.UserMigration)
 resetNeeded, user := runner.SignInUser(ctx, stackOutputs["TableName"],
 stackOutputs["UserPoolClientId"])
 if resetNeeded {
  runner.helper.ListRecentLogEvents(ctx, stackOutputs["MigrateUserFunction"])
  runner.ResetPassword(ctx, stackOutputs["UserPoolClientId"], user)
 }

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函数处理 MigrateUser 触发器。

```
import (
 "context"
 "log"
 "os"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/expression"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
)

const TABLE_NAME = "TABLE_NAME"

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string `dynamodbav:"UserName"`
 UserEmail string `dynamodbav:"UserEmail"`
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the MigrateUser event by looking up a user in an Amazon
 DynamoDB table and
// specifying whether they should be migrated to the user pool.
func (h *handler) HandleRequest(ctx context.Context, event
 events.CognitoEventUserPoolsMigrateUser)
 (events.CognitoEventUserPoolsMigrateUser, error) {
 log.Printf("Received migrate trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 if event.TriggerSource != "UserMigration_Authentication" {
  return event, nil
 }
```

```
tableName := os.Getenv(TABLE_NAME)
user := UserInfo{
 UserName: event.UserName,
}
log.Printf("Looking up user '%v' in table %v.\n", user.UserName, tableName)
filterEx := expression.Name("UserName").Equal(expression.Value(user.UserName))
expr, err := expression.NewBuilder().WithFilter(filterEx).Build()
if err != nil {
 log.Printf("Error building expression to query for user '%v'.\n",
user.UserName)
 return event, err
}
output, err := h.dynamoClient.Scan(ctx, &dynamodb.ScanInput{
 TableName:                 aws.String(tableName),
 FilterExpression:          expr.Filter(),
 ExpressionAttributeNames:  expr.Names(),
 ExpressionAttributeValues: expr.Values(),
})
if err != nil {
 log.Printf("Error looking up user '%v'.\n", user.UserName)
 return event, err
}
if len(output.Items) == 0 {
 log.Printf("User '%v' not found, not migrating user.\n", user.UserName)
 return event, err
}

var users []UserInfo
err = attributevalue.UnmarshalListOfMaps(output.Items, &users)
if err != nil {
 log.Printf("Couldn't unmarshal DynamoDB items. Here's why: %v\n", err)
 return event, err
}

user = users[0]
log.Printf("UserName '%v' found with email %v. User is migrated and must reset
password.\n", user.UserName, user.UserEmail)
event.CognitoEventUserPoolsMigrateUserResponse.UserAttributes =
map[string]string{
 "email":          user.UserEmail,
 "email_verified": "true", // email_verified is required for the forgot password
flow.
}
```

```
 event.CognitoEventUserPoolsMigrateUserResponse.FinalUserStatus =
 "RESET_REQUIRED"
 event.CognitoEventUserPoolsMigrateUserResponse.MessageAction = "SUPPRESS"

 return event, err
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

创建一个执行常见任务的结构。

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
```

```
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
```

```
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
```

```
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int
```

```
const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
```

```
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
```

```
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
```

```
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
```

```
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

创建一个封装 DynamoDB 操作的结构。

```
import (
```

```
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
```

```
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

创建一个封装动作的结构。 Amazon CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API 参考》中的以下主题。
  - ConfirmForgotPassword
  - DeleteUser
  - ForgotPassword
  - InitiateAuth
  - SignUp
  - UpdateUserPool

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用需要使用软件开发工具包进行 MFA 的 Amazon Cognito 用户池注册用户 Amazon

以下代码示例演示了如何：

- 使用用户名、密码和电子邮件地址注册和确认用户。
- 通过将 MFA 应用程序与用户关联来设置多重身份验证。

- 使用密码和 MFA 代码登录。

.NET

适用于 .NET 的 Amazon SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
namespace CognitoBasics;

public class CognitoBasics
{
    private static ILogger logger = null!;

    static async Task Main(string[] args)
    {
        // Set up dependency injection for Amazon Cognito.
        using var host = Host.CreateDefaultBuilder(args)
            .ConfigureLogging(logging =>
                logging.AddFilter("System", LogLevel.Debug)
                    .AddFilter<DebugLoggerProvider>("Microsoft",
 LogLevel.Information)
                    .AddFilter<ConsoleLoggerProvider>("Microsoft",
 LogLevel.Trace))
            .ConfigureServices((_, services) =>
            services.AddAWSService<IAmazonCognitoIdentityProvider>()
            .AddTransient<CognitoWrapper>()
            )
            .Build();

        logger = LoggerFactory.Create(builder => { builder.AddConsole(); })
            .CreateLogger<CognitoBasics>();

        var configuration = new ConfigurationBuilder()
            .SetBasePath(Directory.GetCurrentDirectory())
            .AddJsonFile("settings.json") // Load settings from .json file.
            .AddJsonFile("settings.local.json",
                true) // Optionally load local settings.
```

```
            .Build();

        var cognitoWrapper = host.Services.GetRequiredService<CognitoWrapper>();

        Console.WriteLine(new string('-', 80));
        UiMethods.DisplayOverview();
        Console.WriteLine(new string('-', 80));

        // clientId - The app client Id value that you get from the AWS CDK
script.
        var clientId = configuration["ClientId"]; // "*** REPLACE WITH CLIENT ID
VALUE FROM CDK SCRIPT";

        // poolId - The pool Id that you get from the AWS CDK script.
        var poolId = configuration["PoolId"]!; // "*** REPLACE WITH POOL ID VALUE
FROM CDK SCRIPT";
        var userName = configuration["UserName"];
        var password = configuration["Password"];
        var email = configuration["Email"];

        // If the username wasn't set in the configuration file,
        // get it from the user now.
        if (userName is null)
        {
            do
            {
                Console.Write("Username: ");
                userName = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(userName));
        }
        Console.WriteLine($"\nUsername: {userName}");

        // If the password wasn't set in the configuration file,
        // get it from the user now.
        if (password is null)
        {
            do
            {
                Console.Write("Password: ");
                password = Console.ReadLine();
            }
            while (string.IsNullOrEmpty(password));
        }
```

```
        // If the email address wasn't set in the configuration file,
        // get it from the user now.
        if (email is null)
        {
            do
            {
                Console.Write("Email: ");
                email = Console.ReadLine();
            } while (string.IsNullOrEmpty(email));
        }

        // Now sign up the user.
        Console.WriteLine($"\nSigning up {userName} with email address:
{email}");
        await cognitoWrapper.SignUpAsync(clientId, userName, password, email);

        // Add the user to the user pool.
        Console.WriteLine($"Adding {userName} to the user pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        UiMethods.DisplayTitle("Get confirmation code");
        Console.WriteLine($"Conformation code sent to {userName}.");
        Console.Write("Would you like to send a new code? (Y/N) ");
        var answer = Console.ReadLine();

        if (answer!.ToLower() == "y")
        {
            await cognitoWrapper.ResendConfirmationCodeAsync(clientId, userName);
            Console.WriteLine("Sending a new confirmation code");
        }

        Console.Write("Enter confirmation code (from Email): ");
        var code = Console.ReadLine();

        await cognitoWrapper.ConfirmSignupAsync(clientId, code, userName);

        UiMethods.DisplayTitle("Checking status");
        Console.WriteLine($"Rechecking the status of {userName} in the user
pool");
        await cognitoWrapper.GetAdminUserAsync(userName, poolId);

        Console.WriteLine($"Setting up authenticator for {userName} in the user
pool");
```

```
        var setupResponse = await cognitoWrapper.InitiateAuthAsync(clientId,
 userName, password);

        var setupSession = await
 cognitoWrapper.AssociateSoftwareTokenAsync(setupResponse.Session);
        Console.Write("Enter the 6-digit code displayed in Google Authenticator:
 ");
        var setupCode = Console.ReadLine();

        var setupResult = await
 cognitoWrapper.VerifySoftwareTokenAsync(setupSession, setupCode);
        Console.WriteLine($"Setup status: {setupResult}");

        Console.WriteLine($"Now logging in {userName} in the user pool");
        var authSession = await cognitoWrapper.AdminInitiateAuthAsync(clientId,
 poolId, userName, password);

        Console.Write("Enter a new 6-digit code displayed in Google
 Authenticator: ");
        var authCode = Console.ReadLine();

        var authResult = await
 cognitoWrapper.AdminRespondToAuthChallengeAsync(userName, clientId, authCode,
 authSession, poolId);
        Console.WriteLine($"Authenticated and received access token:
 {authResult.AccessToken}");

        Console.WriteLine(new string('-', 80));
        Console.WriteLine("Cognito scenario is complete.");
        Console.WriteLine(new string('-', 80));
    }
}


using System.Net;

namespace CognitoActions;

/// <summary>
/// Methods to perform Amazon Cognito Identity Provider actions.
/// </summary>
public class CognitoWrapper
{
    private readonly IAmazonCognitoIdentityProvider _cognitoService;
```

```
    /// <summary>
    /// Constructor for the wrapper class containing Amazon Cognito actions.
    /// </summary>
    /// <param name="cognitoService">The Amazon Cognito client object.</param>
    public CognitoWrapper(IAmazonCognitoIdentityProvider cognitoService)
    {
        _cognitoService = cognitoService;
    }

    /// <summary>
    /// List the Amazon Cognito user pools for an account.
    /// </summary>
    /// <returns>A list of UserPoolDescriptionType objects.</returns>
    public async Task<List<UserPoolDescriptionType>> ListUserPoolsAsync()
    {
        var userPools = new List<UserPoolDescriptionType>();

        var userPoolsPaginator = _cognitoService.Paginators.ListUserPools(new
ListUserPoolsRequest());

        await foreach (var response in userPoolsPaginator.Responses)
        {
            userPools.AddRange(response.UserPools);
        }

        return userPools;
    }


    /// <summary>
    /// Get a list of users for the Amazon Cognito user pool.
    /// </summary>
    /// <param name="userPoolId">The user pool ID.</param>
    /// <returns>A list of users.</returns>
    public async Task<List<UserType>> ListUsersAsync(string userPoolId)
    {
        var request = new ListUsersRequest
        {
            UserPoolId = userPoolId
        };

        var users = new List<UserType>();
```

```
        var usersPaginator = _cognitoService.Paginators.ListUsers(request);
        await foreach (var response in usersPaginator.Responses)
        {
            users.AddRange(response.Users);
        }

        return users;
    }


    /// <summary>
    /// Respond to an admin authentication challenge.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="clientId">The client ID.</param>
    /// <param name="mfaCode">The multi-factor authentication code.</param>
    /// <param name="session">The current application session.</param>
    /// <param name="clientId">The user pool ID.</param>
    /// <returns>The result of the authentication response.</returns>
    public async Task<AuthenticationResultType> AdminRespondToAuthChallengeAsync(
        string userName,
        string clientId,
        string mfaCode,
        string session,
        string userPoolId)
    {
        Console.WriteLine("SOFTWARE_TOKEN_MFA challenge is generated");

        var challengeResponses = new Dictionary<string, string>();
        challengeResponses.Add("USERNAME", userName);
        challengeResponses.Add("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        var respondToAuthChallengeRequest = new
AdminRespondToAuthChallengeRequest
        {
            ChallengeName = ChallengeNameType.SOFTWARE_TOKEN_MFA,
            ClientId = clientId,
            ChallengeResponses = challengeResponses,
            Session = session,
            UserPoolId = userPoolId,
        };

        var response = await
_cognitoService.AdminRespondToAuthChallengeAsync(respondToAuthChallengeRequest);
```

```
        Console.WriteLine($"Response to Authentication
{response.AuthenticationResult.TokenType}");
        return response.AuthenticationResult;
    }


    /// <summary>
    /// Verify the TOTP and register for MFA.
    /// </summary>
    /// <param name="session">The name of the session.</param>
    /// <param name="code">The MFA code.</param>
    /// <returns>The status of the software token.</returns>
    public async Task<VerifySoftwareTokenResponseType>
VerifySoftwareTokenAsync(string session, string code)
    {
        var tokenRequest = new VerifySoftwareTokenRequest
        {
            UserCode = code,
            Session = session,
        };

        var verifyResponse = await
_cognitoService.VerifySoftwareTokenAsync(tokenRequest);

        return verifyResponse.Status;
    }


    /// <summary>
    /// Get an MFA token to authenticate the user with the authenticator.
    /// </summary>
    /// <param name="session">The session name.</param>
    /// <returns>The session name.</returns>
    public async Task<string> AssociateSoftwareTokenAsync(string session)
    {
        var softwareTokenRequest = new AssociateSoftwareTokenRequest
        {
            Session = session,
        };

        var tokenResponse = await
_cognitoService.AssociateSoftwareTokenAsync(softwareTokenRequest);
        var secretCode = tokenResponse.SecretCode;
```

```
        Console.WriteLine($"Use the following secret code to set up the
 authenticator: {secretCode}");

        return tokenResponse.Session;
    }


    /// <summary>
    /// Initiate an admin auth request.
    /// </summary>
    /// <param name="clientId">The client ID to use.</param>
    /// <param name="userPoolId">The ID of the user pool.</param>
    /// <param name="userName">The username to authenticate.</param>
    /// <param name="password">The user's password.</param>
    /// <returns>The session to use in challenge-response.</returns>
    public async Task<string> AdminInitiateAuthAsync(string clientId, string
 userPoolId, string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var request = new AdminInitiateAuthRequest
        {
            ClientId = clientId,
            UserPoolId = userPoolId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.AdminInitiateAuthAsync(request);
        return response.Session;
    }

    /// <summary>
    /// Initiate authorization.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The name of the user who is authenticating.</
param>
    /// <param name="password">The password for the user who is authenticating.</
param>
    /// <returns>The response from the initiate auth request.</returns>
```

```
    public async Task<InitiateAuthResponse> InitiateAuthAsync(string clientId,
string userName, string password)
    {
        var authParameters = new Dictionary<string, string>();
        authParameters.Add("USERNAME", userName);
        authParameters.Add("PASSWORD", password);

        var authRequest = new InitiateAuthRequest

        {
            ClientId = clientId,
            AuthParameters = authParameters,
            AuthFlow = AuthFlowType.USER_PASSWORD_AUTH,
        };

        var response = await _cognitoService.InitiateAuthAsync(authRequest);
        Console.WriteLine($"Result Challenge is : {response.ChallengeName}");

        return response;
    }

    /// <summary>
    /// Confirm that the user has signed up.
    /// </summary>
    /// <param name="clientId">The Id of this application.</param>
    /// <param name="code">The confirmation code sent to the user.</param>
    /// <param name="userName">The username.</param>
    /// <returns>True if successful.</returns>
    public async Task<bool> ConfirmSignupAsync(string clientId, string code,
string userName)
    {
        var signUpRequest = new ConfirmSignUpRequest
        {
            ClientId = clientId,
            ConfirmationCode = code,
            Username = userName,
        };

        var response = await _cognitoService.ConfirmSignUpAsync(signUpRequest);
        if (response.HttpStatusCode == HttpStatusCode.OK)
        {
            Console.WriteLine($"{userName} was confirmed");
            return true;
        }
```

```
        return false;
    }


    /// <summary>
    /// Initiates and confirms tracking of the device.
    /// </summary>
    /// <param name="accessToken">The user's access token.</param>
    /// <param name="deviceKey">The key of the device from Amazon Cognito.</
param>
    /// <param name="deviceName">The device name.</param>
    /// <returns></returns>
    public async Task<bool> ConfirmDeviceAsync(string accessToken, string
 deviceKey, string deviceName)
    {
        var request = new ConfirmDeviceRequest
        {
            AccessToken = accessToken,
            DeviceKey = deviceKey,
            DeviceName = deviceName
        };

        var response = await _cognitoService.ConfirmDeviceAsync(request);
        return response.UserConfirmationNecessary;
    }


    /// <summary>
    /// Send a new confirmation code to a user.
    /// </summary>
    /// <param name="clientId">The Id of the client application.</param>
    /// <param name="userName">The username of user who will receive the code.</
param>
    /// <returns>The delivery details.</returns>
    public async Task<CodeDeliveryDetailsType> ResendConfirmationCodeAsync(string
 clientId, string userName)
    {
        var codeRequest = new ResendConfirmationCodeRequest
        {
            ClientId = clientId,
            Username = userName,
        };
```

```
        var response = await
 _cognitoService.ResendConfirmationCodeAsync(codeRequest);

        Console.WriteLine($"Method of delivery is
 {response.CodeDeliveryDetails.DeliveryMedium}");

        return response.CodeDeliveryDetails;
    }


    /// <summary>
    /// Get the specified user from an Amazon Cognito user pool with
 administrator access.
    /// </summary>
    /// <param name="userName">The name of the user.</param>
    /// <param name="poolId">The Id of the Amazon Cognito user pool.</param>
    /// <returns>Async task.</returns>
    public async Task<UserStatusType> GetAdminUserAsync(string userName, string
 poolId)
    {
        AdminGetUserRequest userRequest = new AdminGetUserRequest
        {
            Username = userName,
            UserPoolId = poolId,
        };

        var response = await _cognitoService.AdminGetUserAsync(userRequest);

        Console.WriteLine($"User status {response.UserStatus}");
        return response.UserStatus;
    }


    /// <summary>
    /// Sign up a new user.
    /// </summary>
    /// <param name="clientId">The client Id of the application.</param>
    /// <param name="userName">The username to use.</param>
    /// <param name="password">The user's password.</param>
    /// <param name="email">The email address of the user.</param>
    /// <returns>A Boolean value indicating whether the user was confirmed.</
returns>
    public async Task<bool> SignUpAsync(string clientId, string userName, string
 password, string email)
```

```
    {
        var userAttrs = new AttributeType
        {
            Name = "email",
            Value = email,
        };

        var userAttrsList = new List<AttributeType>();

        userAttrsList.Add(userAttrs);

        var signUpRequest = new SignUpRequest
        {
            UserAttributes = userAttrsList,
            Username = userName,
            ClientId = clientId,
            Password = password
        };

        var response = await _cognitoService.SignUpAsync(signUpRequest);
        return response.HttpStatusCode == HttpStatusCode.OK;
    }

}
```

- 有关 API 详细信息，请参阅《适用于 .NET 的 Amazon SDK API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp

- VerifySoftwareToken

C++

SDK for C++

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
        Aws::Client::ClientConfiguration clientConfig;
        // Optional: Set to the AWS Region (overrides config file).
        // clientConfig.region = "us-east-1";

//! Scenario that adds a user to an Amazon Cognito user pool.
/*!
  \sa gettingStartedWithUserPools()
  \param clientID: Client ID associated with an Amazon Cognito user pool.
  \param userPoolID: An Amazon Cognito user pool ID.
  \param clientConfig: Aws client configuration.
  \return bool: Successful completion.
 */
bool AwsDoc::Cognito::gettingStartedWithUserPools(const Aws::String &clientID,
                                                  const Aws::String &userPoolID,
                                                  const
 Aws::Client::ClientConfiguration &clientConfig) {
    printAsterisksLine();
    std::cout
            << "Welcome to the Amazon Cognito example scenario."
            << std::endl;
    printAsterisksLine();

    std::cout
            << "This scenario will add a user to an Amazon Cognito user pool."
            << std::endl;
    const Aws::String userName = askQuestion("Enter a new username: ");
    const Aws::String password = askQuestion("Enter a new password: ");
    const Aws::String email = askQuestion("Enter a valid email for the user: ");
```

```
   std::cout << "Signing up " << userName << std::endl;

   Aws::CognitoIdentityProvider::CognitoIdentityProviderClient
client(clientConfig);
   bool userExists = false;
   do {
       // 1. Add a user with a username, password, and email address.
       Aws::CognitoIdentityProvider::Model::SignUpRequest request;
       request.AddUserAttributes(
               Aws::CognitoIdentityProvider::Model::AttributeType().WithName(
                       "email").WithValue(email));
       request.SetUsername(userName);
       request.SetPassword(password);
       request.SetClientId(clientID);
       Aws::CognitoIdentityProvider::Model::SignUpOutcome outcome =
               client.SignUp(request);

       if (outcome.IsSuccess()) {
           std::cout << "The signup request for " << userName << " was
successful."
                     << std::endl;
       }
       else if (outcome.GetError().GetErrorType() ==

Aws::CognitoIdentityProvider::CognitoIdentityProviderErrors::USERNAME_EXISTS) {
           std::cout
                   << "The username already exists. Please enter a different
username."
                   << std::endl;
           userExists = true;
       }
       else {
           std::cerr << "Error with CognitoIdentityProvider::SignUpRequest. "
                     << outcome.GetError().GetMessage()
                     << std::endl;
           return false;
       }
   } while (userExists);

   printAsterisksLine();
   std::cout << "Retrieving status of " << userName << " in the user pool."
             << std::endl;
   // 2. Confirm that the user was added to the user pool.
   if (!checkAdminUserStatus(userName, userPoolID, client)) {
```

```
        return false;
    }

    std::cout << "A confirmation code was sent to " << email << "." << std::endl;

    bool resend = askYesNoQuestion("Would you like to send a new code? (y/n) ");
    if (resend) {
        // Request a resend of the confirmation code to the email address.
(ResendConfirmationCode)
        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeRequest
request;
        request.SetUsername(userName);
        request.SetClientId(clientID);

        Aws::CognitoIdentityProvider::Model::ResendConfirmationCodeOutcome
outcome =
                client.ResendConfirmationCode(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "CognitoIdentityProvider::ResendConfirmationCode was
successful."
                    << std::endl;
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::ResendConfirmationCode. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();

    {
        // 4. Send the confirmation code that's received in the email.
(ConfirmSignUp)
        const Aws::String confirmationCode = askQuestion(
                "Enter the confirmation code that was emailed: ");
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpRequest request;
        request.SetClientId(clientID);
        request.SetConfirmationCode(confirmationCode);
        request.SetUsername(userName);
```

```
        Aws::CognitoIdentityProvider::Model::ConfirmSignUpOutcome outcome =
                client.ConfirmSignUp(request);

        if (outcome.IsSuccess()) {
            std::cout << "ConfirmSignup was Successful."
                      << std::endl;
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::ConfirmSignUp. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    std::cout << "Rechecking the status of " << userName << " in the user pool."
              << std::endl;
    if (!checkAdminUserStatus(userName, userPoolID, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout << "Initiating authorization using the username and password."
              << std::endl;

    Aws::String session;
    // 5. Initiate authorization with username and password. (AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID,  userName, password,
 session, client)) {
        return false;
    }

    printAsterisksLine();

    std::cout
            << "Starting setup of time-based one-time password (TOTP) multi-
factor authentication (MFA)."
            << std::endl;

    {
        // 6. Request a setup key for one-time password (TOTP)
        //    multi-factor authentication (MFA). (AssociateSoftwareToken)
```

```
        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenRequest
 request;
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AssociateSoftwareTokenOutcome
 outcome =
                client.AssociateSoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout
                    << "Enter this setup key into an authenticator app, for
 example Google Authenticator."
                    << std::endl;
            std::cout << "Setup key: " << outcome.GetResult().GetSecretCode()
                      << std::endl;
#ifdef USING_QR
            printAsterisksLine();
            std::cout << "\nOr scan the QR code in the file '" << QR_CODE_PATH <<
 "."
                      << std::endl;

            saveQRCode(std::string("otpauth://totp/") + userName + "?secret=" +
                       outcome.GetResult().GetSecretCode());
#endif // USING_QR
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
 CognitoIdentityProvider::AssociateSoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }
    askQuestion("Type enter to continue...", alwaysTrueTest);

    printAsterisksLine();

    {
        Aws::String userCode = askQuestion(
                "Enter the 6 digit code displayed in the authenticator app: ");

        //  7. Send the MFA code copied from an authenticator app.
 (VerifySoftwareToken)
```

```
        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenRequest request;
        request.SetUserCode(userCode);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::VerifySoftwareTokenOutcome outcome =
                client.VerifySoftwareToken(request);

        if (outcome.IsSuccess()) {
            std::cout << "Verification of the code was successful."
                      << std::endl;
            session = outcome.GetResult().GetSession();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::VerifySoftwareToken. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }
    }

    printAsterisksLine();
    std::cout << "You have completed the MFA authentication setup." << std::endl;
    std::cout << "Now, sign in." << std::endl;

    // 8. Initiate authorization again with username and password.
(AdminInitiateAuth)
    if (!adminInitiateAuthorization(clientID, userPoolID, userName, password,
session, client)) {
        return false;
    }

    Aws::String accessToken;
    {
        Aws::String mfaCode = askQuestion(
                "Re-enter the 6 digit code displayed in the authenticator app:
");

        // 9. Send a new MFA code copied from an authenticator app.
(AdminRespondToAuthChallenge)
        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeRequest
request;
        request.AddChallengeResponses("USERNAME", userName);
        request.AddChallengeResponses("SOFTWARE_TOKEN_MFA_CODE", mfaCode);
```

```
        request.SetChallengeName(

Aws::CognitoIdentityProvider::Model::ChallengeNameType::SOFTWARE_TOKEN_MFA);
        request.SetClientId(clientID);
        request.SetUserPoolId(userPoolID);
        request.SetSession(session);

        Aws::CognitoIdentityProvider::Model::AdminRespondToAuthChallengeOutcome
outcome =
                client.AdminRespondToAuthChallenge(request);

        if (outcome.IsSuccess()) {
            std::cout << "Here is the response to the challenge.\n" <<

outcome.GetResult().GetAuthenticationResult().Jsonize().View().WriteReadable()
                      << std::endl;

            accessToken =
outcome.GetResult().GetAuthenticationResult().GetAccessToken();
        }
        else {
            std::cerr << "Error with
CognitoIdentityProvider::AdminRespondToAuthChallenge. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
            return false;
        }

        std::cout << "You have successfully added a user to Amazon Cognito."
                  << std::endl;
    }

    if (askYesNoQuestion("Would you like to delete the user that you just added?
(y/n) ")) {
        // 10. Delete the user that you just added. (DeleteUser)
        Aws::CognitoIdentityProvider::Model::DeleteUserRequest request;
        request.SetAccessToken(accessToken);

        Aws::CognitoIdentityProvider::Model::DeleteUserOutcome outcome =
                client.DeleteUser(request);

        if (outcome.IsSuccess()) {
            std::cout << "The user " << userName << " was deleted."
                      << std::endl;
```

```
        }
        else {
            std::cerr << "Error with CognitoIdentityProvider::DeleteUser. "
                      << outcome.GetError().GetMessage()
                      << std::endl;
        }
    }

    return true;
}

//! Routine which checks the user status in an Amazon Cognito user pool.
/*!
 \sa checkAdminUserStatus()
 \param userName: A username.
 \param userPoolID: An Amazon Cognito user pool ID.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::checkAdminUserStatus(const Aws::String &userName,
                                           const Aws::String &userPoolID,
                                           const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminGetUserRequest request;
    request.SetUsername(userName);
    request.SetUserPoolId(userPoolID);

    Aws::CognitoIdentityProvider::Model::AdminGetUserOutcome outcome =
            client.AdminGetUser(request);

    if (outcome.IsSuccess()) {
        std::cout << "The status for " << userName << " is " <<

 Aws::CognitoIdentityProvider::Model::UserStatusTypeMapper::GetNameForUserStatusType(
                        outcome.GetResult().GetUserStatus()) << std::endl;
        std::cout << "Enabled is " << outcome.GetResult().GetEnabled() <<
 std::endl;
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminGetUser. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}

//! Routine which starts authorization of an Amazon Cognito user.
//! This routine requires administrator credentials.
/*!
 \sa adminInitiateAuthorization()
 \param clientID: Client ID of tracked device.
 \param userPoolID: An Amazon Cognito user pool ID.
 \param userName: A username.
 \param password: A password.
 \param sessionResult: String to receive a session token.
 \return bool: Successful completion.
 */
bool AwsDoc::Cognito::adminInitiateAuthorization(const Aws::String &clientID,
                                                 const Aws::String &userPoolID,
                                                 const Aws::String &userName,
                                                 const Aws::String &password,
                                                 Aws::String &sessionResult,
                                                 const
 Aws::CognitoIdentityProvider::CognitoIdentityProviderClient &client) {
    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthRequest request;
    request.SetClientId(clientID);
    request.SetUserPoolId(userPoolID);
    request.AddAuthParameters("USERNAME", userName);
    request.AddAuthParameters("PASSWORD", password);
    request.SetAuthFlow(

 Aws::CognitoIdentityProvider::Model::AuthFlowType::ADMIN_USER_PASSWORD_AUTH);


    Aws::CognitoIdentityProvider::Model::AdminInitiateAuthOutcome outcome =
            client.AdminInitiateAuth(request);

    if (outcome.IsSuccess()) {
        std::cout << "Call to AdminInitiateAuth was successful." << std::endl;
        sessionResult = outcome.GetResult().GetSession();
    }
    else {
        std::cerr << "Error with CognitoIdentityProvider::AdminInitiateAuth. "
                  << outcome.GetError().GetMessage()
                  << std::endl;
    }

    return outcome.IsSuccess();
```

```
}
```

- 有关 API 详细信息，请参阅《适用于 C++ 的 Amazon SDK API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Java

适用于 Java 的 SDK 2.x

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库 中查找完整示例，了解如何进行设置和运行。

```
import software.amazon.awssdk.regions.Region;
import
 software.amazon.awssdk.services.cognitoidentityprovider.CognitoIdentityProviderClient;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminGetUserResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthRequest;
```

```
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminInitiateAuthResponse;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AdminRespondToAuthChalleng
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AssociateSoftwareTokenResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AttributeType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.AuthFlowType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ChallengeNameType;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.CognitoIdentityProviderExc
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ConfirmSignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeRequ
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.ResendConfirmationCodeResp
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.SignUpRequest;
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRequest
import
 software.amazon.awssdk.services.cognitoidentityprovider.model.VerifySoftwareTokenRespons
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import java.util.Scanner;

/**
 * Before running this Java V2 code example, set up your development
 * environment, including your credentials.
 *
 * For more information, see the following documentation:
 *
```

```
 * https://docs.aws.amazon.com/sdk-for-java/latest/developer-guide/get-
started.html
 *
 * TIP: To set up the required user pool, run the AWS Cloud Development Kit (AWS
 * CDK) script provided in this GitHub repo at
 * resources/cdk/cognito_scenario_user_pool_with_mfa.
 *
 * This code example performs the following operations:
 *
 * 1. Invokes the signUp method to sign up a user.
 * 2. Invokes the adminGetUser method to get the user's confirmation status.
 * 3. Invokes the ResendConfirmationCode method if the user requested another
 * code.
 * 4. Invokes the confirmSignUp method.
 * 5. Invokes the AdminInitiateAuth to sign in. This results in being prompted
 * to set up TOTP (time-based one-time password). (The response is
 * “ChallengeName”: “MFA_SETUP”).
 * 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
 * key. This can be used with Google Authenticator.
 * 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 * MFA.
 * 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 * prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 * 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

public class CognitoMVP {
    public static final String DASHES = new String(new char[80]).replace("\0",
 "-");

    public static void main(String[] args) throws NoSuchAlgorithmException,
 InvalidKeyException {
        final String usage = """

                Usage:
                    <clientId> <poolId>

                Where:
                    clientId - The app client Id value that you can get from the
 AWS CDK script.
                    poolId - The pool Id that you can get from the AWS CDK
 script.\s
                """;
```

```
        if (args.length != 2) {
            System.out.println(usage);
            System.exit(1);
        }

        String clientId = args[0];
        String poolId = args[1];
        CognitoIdentityProviderClient identityProviderClient =
CognitoIdentityProviderClient.builder()
                .region(Region.US_EAST_1)
                .build();

        System.out.println(DASHES);
        System.out.println("Welcome to the Amazon Cognito example scenario.");
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter your user name");
        Scanner in = new Scanner(System.in);
        String userName = in.nextLine();

        System.out.println("*** Enter your password");
        String password = in.nextLine();

        System.out.println("*** Enter your email");
        String email = in.nextLine();

        System.out.println("1. Signing up " + userName);
        signUp(identityProviderClient, clientId, userName, password, email);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("2. Getting " + userName + " in the user pool");
        getAdminUser(identityProviderClient, userName, poolId);

        System.out
                .println("*** Conformation code sent to " + userName + ". Would
you like to send a new code? (Yes/No)");
        System.out.println(DASHES);

        System.out.println(DASHES);
        String ans = in.nextLine();

        if (ans.compareTo("Yes") == 0) {
```

```
            resendConfirmationCode(identityProviderClient, clientId, userName);
            System.out.println("3. Sending a new confirmation code");
        }
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("4. Enter confirmation code that was emailed");
        String code = in.nextLine();
        confirmSignUp(identityProviderClient, clientId, code, userName);
        System.out.println("Rechecking the status of " + userName + " in the user
pool");
        getAdminUser(identityProviderClient, userName, poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("5. Invokes the initiateAuth to sign in");
        AdminInitiateAuthResponse authResponse =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        String mySession = authResponse.session();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("6. Invokes the AssociateSoftwareToken method to
generate a TOTP key");
        String newSession = getSecretForAppMFA(identityProviderClient,
mySession);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("*** Enter the 6-digit code displayed in Google
Authenticator");
        String myCode = in.nextLine();
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("7. Verify the TOTP and register for MFA");
        verifyTOTP(identityProviderClient, newSession, myCode);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("8. Re-enter a 6-digit code displayed in Google
Authenticator");
        String mfaCode = in.nextLine();
```

```
        AdminInitiateAuthResponse authResponse1 =
initiateAuth(identityProviderClient, clientId, userName, password,
                poolId);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("9.  Invokes the AdminRespondToAuthChallenge");
        String session2 = authResponse1.session();
        adminRespondToAuthChallenge(identityProviderClient, userName, clientId,
mfaCode, session2);
        System.out.println(DASHES);

        System.out.println(DASHES);
        System.out.println("All Amazon Cognito operations were successfully
performed");
        System.out.println(DASHES);
    }

    // Respond to an authentication challenge.
    public static void adminRespondToAuthChallenge(CognitoIdentityProviderClient
identityProviderClient,
            String userName, String clientId, String mfaCode, String session) {
        System.out.println("SOFTWARE_TOKEN_MFA challenge is generated");
        Map<String, String> challengeResponses = new HashMap<>();

        challengeResponses.put("USERNAME", userName);
        challengeResponses.put("SOFTWARE_TOKEN_MFA_CODE", mfaCode);

        AdminRespondToAuthChallengeRequest respondToAuthChallengeRequest =
AdminRespondToAuthChallengeRequest.builder()
                .challengeName(ChallengeNameType.SOFTWARE_TOKEN_MFA)
                .clientId(clientId)
                .challengeResponses(challengeResponses)
                .session(session)
                .build();

        AdminRespondToAuthChallengeResponse respondToAuthChallengeResult =
identityProviderClient
                .adminRespondToAuthChallenge(respondToAuthChallengeRequest);

System.out.println("respondToAuthChallengeResult.getAuthenticationResult()"
                + respondToAuthChallengeResult.authenticationResult());
    }
```

```
   // Verify the TOTP and register for MFA.
   public static void verifyTOTP(CognitoIdentityProviderClient
identityProviderClient, String session, String code) {
       try {
           VerifySoftwareTokenRequest tokenRequest =
VerifySoftwareTokenRequest.builder()
                   .userCode(code)
                   .session(session)
                   .build();

           VerifySoftwareTokenResponse verifyResponse =
identityProviderClient.verifySoftwareToken(tokenRequest);
           System.out.println("The status of the token is " +
verifyResponse.statusAsString());

       } catch (CognitoIdentityProviderException e) {
           System.err.println(e.awsErrorDetails().errorMessage());
           System.exit(1);
       }
   }

   public static AdminInitiateAuthResponse
initiateAuth(CognitoIdentityProviderClient identityProviderClient,
           String clientId, String userName, String password, String userPoolId)
{
       try {
           Map<String, String> authParameters = new HashMap<>();
           authParameters.put("USERNAME", userName);
           authParameters.put("PASSWORD", password);

           AdminInitiateAuthRequest authRequest =
AdminInitiateAuthRequest.builder()
                   .clientId(clientId)
                   .userPoolId(userPoolId)
                   .authParameters(authParameters)
                   .authFlow(AuthFlowType.ADMIN_USER_PASSWORD_AUTH)
                   .build();

           AdminInitiateAuthResponse response =
identityProviderClient.adminInitiateAuth(authRequest);
           System.out.println("Result Challenge is : " +
response.challengeName());
           return response;
```

```
        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }

        return null;
    }

    public static String getSecretForAppMFA(CognitoIdentityProviderClient
identityProviderClient, String session) {
        AssociateSoftwareTokenRequest softwareTokenRequest =
AssociateSoftwareTokenRequest.builder()
                .session(session)
                .build();

        AssociateSoftwareTokenResponse tokenResponse = identityProviderClient
                .associateSoftwareToken(softwareTokenRequest);
        String secretCode = tokenResponse.secretCode();
        System.out.println("Enter this token into Google Authenticator");
        System.out.println(secretCode);
        return tokenResponse.session();
    }

    public static void confirmSignUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String code,
            String userName) {
        try {
            ConfirmSignUpRequest signUpRequest = ConfirmSignUpRequest.builder()
                    .clientId(clientId)
                    .confirmationCode(code)
                    .username(userName)
                    .build();

            identityProviderClient.confirmSignUp(signUpRequest);
            System.out.println(userName + " was confirmed");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void resendConfirmationCode(CognitoIdentityProviderClient
identityProviderClient, String clientId,
```

```
            String userName) {
        try {
            ResendConfirmationCodeRequest codeRequest =
ResendConfirmationCodeRequest.builder()
                    .clientId(clientId)
                    .username(userName)
                    .build();

            ResendConfirmationCodeResponse response =
identityProviderClient.resendConfirmationCode(codeRequest);
            System.out.println("Method of delivery is " +
response.codeDeliveryDetails().deliveryMediumAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }

    public static void signUp(CognitoIdentityProviderClient
identityProviderClient, String clientId, String userName,
            String password, String email) {
        AttributeType userAttrs = AttributeType.builder()
                .name("email")
                .value(email)
                .build();

        List<AttributeType> userAttrsList = new ArrayList<>();
        userAttrsList.add(userAttrs);
        try {
            SignUpRequest signUpRequest = SignUpRequest.builder()
                    .userAttributes(userAttrsList)
                    .username(userName)
                    .clientId(clientId)
                    .password(password)
                    .build();

            identityProviderClient.signUp(signUpRequest);
            System.out.println("User has been signed up ");

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
```

```
    }

    public static void getAdminUser(CognitoIdentityProviderClient
 identityProviderClient, String userName,
            String poolId) {
        try {
            AdminGetUserRequest userRequest = AdminGetUserRequest.builder()
                    .username(userName)
                    .userPoolId(poolId)
                    .build();

            AdminGetUserResponse response =
 identityProviderClient.adminGetUser(userRequest);
            System.out.println("User status " + response.userStatusAsString());

        } catch (CognitoIdentityProviderException e) {
            System.err.println(e.awsErrorDetails().errorMessage());
            System.exit(1);
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Java 2.x API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

JavaScript

适用于 JavaScript (v3) 的软件开发工具包

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

为了获得最佳体验，请克隆 GitHub 存储库并运行此示例。以下代码代表完整示例应用程序的示例。

```
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { signUp } from "../../../actions/sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateUser = (username, password, email) => {
  if (!(username && password && email)) {
    throw new Error(
      `Username, password, and email must be provided as arguments to the 'sign-
up' command.`,
    );
  }
};

const signUpHandler = async (commands) => {
  const [_, username, password, email] = commands;

  try {
    validateUser(username, password, email);
    /**
```

```
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Signing up.");
    await signUp({ clientId, username, password, email });
    logger.log(`Signed up. A confirmation email has been sent to: ${email}.`);
    logger.log(
      `Run 'confirm-sign-up ${username} <code>' to confirm your account.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { signUpHandler };

const signUp = ({ clientId, username, password, email }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new SignUpCommand({
    ClientId: clientId,
    Username: username,
    Password: password,
    UserAttributes: [{ Name: "email", Value: email }],
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { confirmSignUp } from "../../../actions/confirm-sign-up.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getSecondValuesFromEntries } from "@aws-doc-sdk-examples/lib/utils/util-
csv.js";

const validateClient = (clientId) => {
  if (!clientId) {
    throw new Error(
      `App client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};
```

```
const validateUser = (username) => {
  if (!username) {
    throw new Error(
      `Username name is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const validateCode = (code) => {
  if (!code) {
    throw new Error(
      `Verification code is missing. It must be provided as an argument to the
 'confirm-sign-up' command.`,
    );
  }
};

const confirmSignUpHandler = async (commands) => {
  const [_, username, code] = commands;

  try {
    validateUser(username);
    validateCode(code);
    /**
     * @type {string[]}
     */
    const values = getSecondValuesFromEntries(FILE_USER_POOLS);
    const clientId = values[0];
    validateClient(clientId);
    logger.log("Confirming user.");
    await confirmSignUp({ clientId, username, code });
    logger.log(
      `User confirmed. Run 'admin-initiate-auth ${username} <password>' to sign
 in.`,
    );
  } catch (err) {
    logger.error(err);
  }
};

export { confirmSignUpHandler };
```

```
const confirmSignUp = ({ clientId, username, code }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new ConfirmSignUpCommand({
    ClientId: clientId,
    Username: username,
    ConfirmationCode: code,
  });

  return client.send(command);
};

import qrcode from "qrcode-terminal";
import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminInitiateAuth } from "../../../actions/admin-initiate-auth.js";
import { associateSoftwareToken } from "../../../actions/associate-software-
token.js";
import { FILE_USER_POOLS } from "./constants.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";

const handleMfaSetup = async (session, username) => {
  const { SecretCode, Session } = await associateSoftwareToken(session);

  // Store the Session for use with 'VerifySoftwareToken'.
  process.env.SESSION = Session;

  console.log(
    "Scan this code in your preferred authenticator app, then run 'verify-
software-token' to finish the setup.",
  );
  qrcode.generate(
    `otpauth://totp/${username}?secret=${SecretCode}`,
    { small: true },
    console.log,
  );
};

const handleSoftwareTokenMfa = (session) => {
  // Store the Session for use with 'AdminRespondToAuthChallenge'.
  process.env.SESSION = session;
};

const validateClient = (id) => {
  if (!id) {
```

```
    throw new Error(
      `User pool client id is missing. Did you run 'create-user-pool'?`,
    );
  }
};

const validateId = (id) => {
  if (!id) {
    throw new Error(`User pool id is missing. Did you run 'create-user-pool'?`);
  }
};

const validateUser = (username, password) => {
  if (!(username && password)) {
    throw new Error(
      `Username and password must be provided as arguments to the 'admin-
initiate-auth' command.`,
    );
  }
};

const adminInitiateAuthHandler = async (commands) => {
  const [_, username, password] = commands;

  try {
    validateUser(username, password);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    validateId(userPoolId);
    validateClient(clientId);

    logger.log("Signing in.");
    const { ChallengeName, Session } = await adminInitiateAuth({
      clientId,
      userPoolId,
      username,
      password,
    });

    if (ChallengeName === "MFA_SETUP") {
      logger.log("MFA setup is required.");
      return handleMfaSetup(Session, username);
    }
```

```
    if (ChallengeName === "SOFTWARE_TOKEN_MFA") {
      handleSoftwareTokenMfa(Session);
      logger.log(`Run 'admin-respond-to-auth-challenge ${username} <totp>'`);
    }
  } catch (err) {
    logger.error(err);
  }
};

export { adminInitiateAuthHandler };

const adminInitiateAuth = ({ clientId, userPoolId, username, password }) => {
  const client = new CognitoIdentityProviderClient({});

  const command = new AdminInitiateAuthCommand({
    ClientId: clientId,
    UserPoolId: userPoolId,
    AuthFlow: AuthFlowType.ADMIN_USER_PASSWORD_AUTH,
    AuthParameters: { USERNAME: username, PASSWORD: password },
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { adminRespondToAuthChallenge } from "../../../actions/admin-respond-to-
auth-challenge.js";
import { getFirstEntry } from "@aws-doc-sdk-examples/lib/utils/util-csv.js";
import { FILE_USER_POOLS } from "./constants.js";

const verifyUsername = (username) => {
  if (!username) {
    throw new Error(
      `Username is missing. It must be provided as an argument to the 'admin-
respond-to-auth-challenge' command.`,
    );
  }
};

const verifyTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) is missing. It must be provided as an
 argument to the 'admin-respond-to-auth-challenge' command.`,
```

```
    );
  }
};

const storeAccessToken = (token) => {
  process.env.AccessToken = token;
};

const adminRespondToAuthChallengeHandler = async (commands) => {
  const [_, username, totp] = commands;

  try {
    verifyUsername(username);
    verifyTotp(totp);

    const [userPoolId, clientId] = getFirstEntry(FILE_USER_POOLS);
    const session = process.env.SESSION;

    const { AuthenticationResult } = await adminRespondToAuthChallenge({
      clientId,
      userPoolId,
      username,
      totp,
      session,
    });

    storeAccessToken(AuthenticationResult.AccessToken);

    logger.log("Successfully authenticated.");
  } catch (err) {
    logger.error(err);
  }
};

export { adminRespondToAuthChallengeHandler };

const respondToAuthChallenge = ({
  clientId,
  username,
  session,
  userPoolId,
  code,
}) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  const command = new RespondToAuthChallengeCommand({
    ChallengeName: ChallengeNameType.SOFTWARE_TOKEN_MFA,
    ChallengeResponses: {
      SOFTWARE_TOKEN_MFA_CODE: code,
      USERNAME: username,
    },
    ClientId: clientId,
    UserPoolId: userPoolId,
    Session: session,
  });

  return client.send(command);
};

import { logger } from "@aws-doc-sdk-examples/lib/utils/util-log.js";
import { verifySoftwareToken } from "../../../actions/verify-software-token.js";

const validateTotp = (totp) => {
  if (!totp) {
    throw new Error(
      `Time-based one-time password (TOTP) must be provided to the 'validate-
software-token' command.`,
    );
  }
};
const verifySoftwareTokenHandler = async (commands) => {
  const [_, totp] = commands;

  try {
    validateTotp(totp);

    logger.log("Verifying TOTP.");
    await verifySoftwareToken(totp);
    logger.log("TOTP Verified. Run 'admin-initiate-auth' again to sign-in.");
  } catch (err) {
    logger.error(err);
  }
};

export { verifySoftwareTokenHandler };

const verifySoftwareToken = (totp) => {
  const client = new CognitoIdentityProviderClient({});
```

```
  // The 'Session' is provided in the response to 'AssociateSoftwareToken'.
  const session = process.env.SESSION;

  if (!session) {
    throw new Error(
      "Missing a valid Session. Did you run 'admin-initiate-auth'?",
    );
  }

  const command = new VerifySoftwareTokenCommand({
    Session: session,
    UserCode: totp,
  });

  return client.send(command);
};
```

- 有关 API 详细信息，请参阅《适用于 JavaScript 的 Amazon SDK API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Kotlin

适用于 Kotlin 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

```
/**
 Before running this Kotlin code example, set up your development environment,
 including your credentials.

 For more information, see the following documentation:
 https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html

 TIP: To set up the required user pool, run the AWS Cloud Development
 Kit (AWS CDK) script provided in this GitHub repo at resources/cdk/
cognito_scenario_user_pool_with_mfa.

 This code example performs the following operations:

 1. Invokes the signUp method to sign up a user.
 2. Invokes the adminGetUser method to get the user's confirmation status.
 3. Invokes the ResendConfirmationCode method if the user requested another code.
 4. Invokes the confirmSignUp method.
 5. Invokes the initiateAuth to sign in. This results in being prompted to
 set up TOTP (time-based one-time password). (The response is “ChallengeName”:
 “MFA_SETUP”).
 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private key.
 This can be used with Google Authenticator.
 7. Invokes the VerifySoftwareToken method to verify the TOTP and register for
 MFA.
 8. Invokes the AdminInitiateAuth to sign in again. This results in being
 prompted to submit a TOTP (Response: “ChallengeName”: “SOFTWARE_TOKEN_MFA”).
 9. Invokes the AdminRespondToAuthChallenge to get back a token.
 */

suspend fun main(args: Array<String>) {
    val usage = """
```

```
        Usage:
            <clientId> <poolId>
        Where:
            clientId - The app client Id value that you can get from the AWS CDK
script.
            poolId - The pool Id that you can get from the AWS CDK script.
    """

    if (args.size != 2) {
        println(usage)
        exitProcess(1)
    }

    val clientId = args[0]
    val poolId = args[1]

    // Use the console to get data from the user.
    println("*** Enter your use name")
    val inOb = Scanner(System.`in`)
    val userName = inOb.nextLine()
    println(userName)

    println("*** Enter your password")
    val password: String = inOb.nextLine()

    println("*** Enter your email")
    val email = inOb.nextLine()

    println("*** Signing up $userName")
    signUp(clientId, userName, password, email)

    println("*** Getting $userName in the user pool")
    getAdminUser(userName, poolId)

    println("*** Conformation code sent to $userName. Would you like to send a
new code? (Yes/No)")
    val ans = inOb.nextLine()

    if (ans.compareTo("Yes") == 0) {
        println("*** Sending a new confirmation code")
        resendConfirmationCode(clientId, userName)
    }
    println("*** Enter the confirmation code that was emailed")
    val code = inOb.nextLine()
```

```
    confirmSignUp(clientId, code, userName)

    println("*** Rechecking the status of $userName in the user pool")
    getAdminUser(userName, poolId)

    val authResponse = checkAuthMethod(clientId, userName, password, poolId)
    val mySession = authResponse.session
    val newSession = getSecretForAppMFA(mySession)
    println("*** Enter the 6-digit code displayed in Google Authenticator")
    val myCode = inOb.nextLine()

    // Verify the TOTP and register for MFA.
    verifyTOTP(newSession, myCode)
    println("*** Re-enter a 6-digit code displayed in Google Authenticator")
    val mfaCode: String = inOb.nextLine()
    val authResponse1 = checkAuthMethod(clientId, userName, password, poolId)
    val session2 = authResponse1.session
    adminRespondToAuthChallenge(userName, clientId, mfaCode, session2)
}

suspend fun checkAuthMethod(
    clientIdVal: String,
    userNameVal: String,
    passwordVal: String,
    userPoolIdVal: String,
): AdminInitiateAuthResponse {
    val authParas = mutableMapOf<String, String>()
    authParas["USERNAME"] = userNameVal
    authParas["PASSWORD"] = passwordVal

    val authRequest =
        AdminInitiateAuthRequest {
            clientId = clientIdVal
            userPoolId = userPoolIdVal
            authParameters = authParas
            authFlow = AuthFlowType.AdminUserPasswordAuth
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminInitiateAuth(authRequest)
        println("Result Challenge is ${response.challengeName}")
        return response
    }
```

```
}

suspend fun resendConfirmationCode(
    clientIdVal: String?,
    userNameVal: String?,
) {
    val codeRequest =
        ResendConfirmationCodeRequest {
            clientId = clientIdVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.resendConfirmationCode(codeRequest)
        println("Method of delivery is " +
 (response.codeDeliveryDetails?.deliveryMedium))
    }
}

// Respond to an authentication challenge.
suspend fun adminRespondToAuthChallenge(
    userName: String,
    clientIdVal: String?,
    mfaCode: String,
    sessionVal: String?,
) {
    println("SOFTWARE_TOKEN_MFA challenge is generated")
    val challengeResponsesOb = mutableMapOf<String, String>()
    challengeResponsesOb["USERNAME"] = userName
    challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

    val adminRespondToAuthChallengeRequest =
        AdminRespondToAuthChallengeRequest {
            challengeName = ChallengeNameType.SoftwareTokenMfa
            clientId = clientIdVal
            challengeResponses = challengeResponsesOb
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val respondToAuthChallengeResult =
 identityProviderClient.adminRespondToAuthChallenge(adminRespondToAuthChallengeRequest)
```

```
        println("respondToAuthChallengeResult.getAuthenticationResult()
 ${respondToAuthChallengeResult.authenticationResult}")
    }
}

// Verify the TOTP and register for MFA.
suspend fun verifyTOTP(
    sessionVal: String?,
    codeVal: String?,
) {
    val tokenRequest =
        VerifySoftwareTokenRequest {
            userCode = codeVal
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val verifyResponse =
 identityProviderClient.verifySoftwareToken(tokenRequest)
        println("The status of the token is ${verifyResponse.status}")
    }
}

suspend fun getSecretForAppMFA(sessionVal: String?): String? {
    val softwareTokenRequest =
        AssociateSoftwareTokenRequest {
            session = sessionVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val tokenResponse =
 identityProviderClient.associateSoftwareToken(softwareTokenRequest)
        val secretCode = tokenResponse.secretCode
        println("Enter this token into Google Authenticator")
        println(secretCode)
        return tokenResponse.session
    }
}

suspend fun confirmSignUp(
    clientIdVal: String?,
    codeVal: String?,
```

```
    userNameVal: String?,
) {
    val signUpRequest =
        ConfirmSignUpRequest {
            clientId = clientIdVal
            confirmationCode = codeVal
            username = userNameVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.confirmSignUp(signUpRequest)
        println("$userNameVal  was confirmed")
    }
}

suspend fun getAdminUser(
    userNameVal: String?,
    poolIdVal: String?,
) {
    val userRequest =
        AdminGetUserRequest {
            username = userNameVal
            userPoolId = poolIdVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        val response = identityProviderClient.adminGetUser(userRequest)
        println("User status ${response.userStatus}")
    }
}

suspend fun signUp(
    clientIdVal: String?,
    userNameVal: String?,
    passwordVal: String?,
    emailVal: String?,
) {
    val userAttrs =
        AttributeType {
            name = "email"
            value = emailVal
        }
```

```
    val userAttrsList = mutableListOf<AttributeType>()
    userAttrsList.add(userAttrs)
    val signUpRequest =
        SignUpRequest {
            userAttributes = userAttrsList
            username = userNameVal
            clientId = clientIdVal
            password = passwordVal
        }

    CognitoIdentityProviderClient { region = "us-east-1" }.use
 { identityProviderClient ->
        identityProviderClient.signUp(signUpRequest)
        println("User has been signed up")
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Kotlin API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Python

适用于 Python 的 SDK（Boto3）

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

创建一个包含场景中使用的 Amazon Cognito 函数的类。

```
class CognitoIdentityProviderWrapper:
    """Encapsulates Amazon Cognito actions"""

    def __init__(self, cognito_idp_client, user_pool_id, client_id,
 client_secret=None):
        """
        :param cognito_idp_client: A Boto3 Amazon Cognito Identity Provider
 client.
        :param user_pool_id: The ID of an existing Amazon Cognito user pool.
        :param client_id: The ID of a client application registered with the user
 pool.
        :param client_secret: The client secret, if the client has a secret.
        """
        self.cognito_idp_client = cognito_idp_client
        self.user_pool_id = user_pool_id
        self.client_id = client_id
        self.client_secret = client_secret


    def _secret_hash(self, user_name):
        """
        Calculates a secret hash from a user name and a client secret.

        :param user_name: The user name to use when calculating the hash.
        :return: The secret hash.
        """
        key = self.client_secret.encode()
        msg = bytes(user_name + self.client_id, "utf-8")
        secret_hash = base64.b64encode(
            hmac.new(key, msg, digestmod=hashlib.sha256).digest()
        ).decode()
```

```
        logger.info("Made secret hash for %s: %s.", user_name, secret_hash)
        return secret_hash

    def sign_up_user(self, user_name, password, user_email):
        """
        Signs up a new user with Amazon Cognito. This action prompts Amazon
Cognito
        to send an email to the specified email address. The email contains a
code that
        can be used to confirm the user.

        When the user already exists, the user status is checked to determine
whether
        the user has been confirmed.

        :param user_name: The user name that identifies the new user.
        :param password: The password for the new user.
        :param user_email: The email address for the new user.
        :return: True when the user is already confirmed with Amazon Cognito.
                 Otherwise, false.
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "Password": password,
                "UserAttributes": [{"Name": "email", "Value": user_email}],
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.sign_up(**kwargs)
            confirmed = response["UserConfirmed"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "UsernameExistsException":
                response = self.cognito_idp_client.admin_get_user(
                    UserPoolId=self.user_pool_id, Username=user_name
                )
                logger.warning(
                    "User %s exists and is %s.", user_name,
response["UserStatus"]
                )
                confirmed = response["UserStatus"] == "CONFIRMED"
            else:
                logger.error(
```

```
                    "Couldn't sign up %s. Here's why: %s: %s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        return confirmed


    def resend_confirmation(self, user_name):
        """
        Prompts Amazon Cognito to resend an email with a new confirmation code.

        :param user_name: The name of the user who will receive the email.
        :return: Delivery information about where the email is sent.
        """
        try:
            kwargs = {"ClientId": self.client_id, "Username": user_name}
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            response = self.cognito_idp_client.resend_confirmation_code(**kwargs)
            delivery = response["CodeDeliveryDetails"]
        except ClientError as err:
            logger.error(
                "Couldn't resend confirmation to %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return delivery


    def confirm_user_sign_up(self, user_name, confirmation_code):
        """
        Confirms a previously created user. A user must be confirmed before they
        can sign in to Amazon Cognito.

        :param user_name: The name of the user to confirm.
        :param confirmation_code: The confirmation code sent to the user's
registered
                                  email address.
        :return: True when the confirmation succeeds.
```

```
        """
        try:
            kwargs = {
                "ClientId": self.client_id,
                "Username": user_name,
                "ConfirmationCode": confirmation_code,
            }
            if self.client_secret is not None:
                kwargs["SecretHash"] = self._secret_hash(user_name)
            self.cognito_idp_client.confirm_sign_up(**kwargs)
        except ClientError as err:
            logger.error(
                "Couldn't confirm sign up for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return True


    def list_users(self):
        """
        Returns a list of the users in the current user pool.

        :return: The list of users.
        """
        try:
            response =
self.cognito_idp_client.list_users(UserPoolId=self.user_pool_id)
            users = response["Users"]
        except ClientError as err:
            logger.error(
                "Couldn't list users for %s. Here's why: %s: %s",
                self.user_pool_id,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return users
```

```
    def start_sign_in(self, user_name, password):
        """
        Starts the sign-in process for a user by using administrator credentials.
        This method of signing in is appropriate for code running on a secure
 server.

        If the user pool is configured to require MFA and this is the first sign-
in
        for the user, Amazon Cognito returns a challenge response to set up an
        MFA application. When this occurs, this function gets an MFA secret from
        Amazon Cognito and returns it to the caller.

        :param user_name: The name of the user to sign in.
        :param password: The user's password.
        :return: The result of the sign-in attempt. When sign-in is successful,
 this
                 returns an access token that can be used to get AWS credentials.
 Otherwise,
                 Amazon Cognito returns a challenge to set up an MFA application,
                 or a challenge to enter an MFA code from a registered MFA
 application.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
                "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
                "AuthParameters": {"USERNAME": user_name, "PASSWORD": password},
            }
            if self.client_secret is not None:
                kwargs["AuthParameters"]["SECRET_HASH"] =
 self._secret_hash(user_name)
            response = self.cognito_idp_client.admin_initiate_auth(**kwargs)
            challenge_name = response.get("ChallengeName", None)
            if challenge_name == "MFA_SETUP":
                if (
                    "SOFTWARE_TOKEN_MFA"
                    in response["ChallengeParameters"]["MFAS_CAN_SETUP"]
                ):
                    response.update(self.get_mfa_secret(response["Session"]))
                else:
                    raise RuntimeError(
                        "The user pool requires MFA setup, but the user pool is
 not "
```

```
                        "configured for TOTP MFA. This example requires TOTP
MFA."
                    )
        except ClientError as err:
            logger.error(
                "Couldn't start sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def get_mfa_secret(self, session):
        """
        Gets a token that can be used to associate an MFA application with the
user.

        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :return: An MFA token that can be used to set up an MFA application.
        """
        try:
            response =
self.cognito_idp_client.associate_software_token(Session=session)
        except ClientError as err:
            logger.error(
                "Couldn't get MFA secret. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def verify_mfa(self, session, user_code):
        """
        Verify a new MFA application that is associated with a user.
```

```
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param user_code: A code generated by the associated MFA application.
        :return: Status that indicates whether the MFA application is verified.
        """
        try:
            response = self.cognito_idp_client.verify_software_token(
                Session=session, UserCode=user_code
            )
        except ClientError as err:
            logger.error(
                "Couldn't verify MFA. Here's why: %s: %s",
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            response.pop("ResponseMetadata", None)
            return response


    def respond_to_mfa_challenge(self, user_name, session, mfa_code):
        """
        Responds to a challenge for an MFA code. This completes the second step
of
        a two-factor sign-in. When sign-in is successful, it returns an access
token
        that can be used to get AWS credentials from Amazon Cognito.

        :param user_name: The name of the user who is signing in.
        :param session: Session information returned from a previous call to
initiate
                        authentication.
        :param mfa_code: A code generated by the associated MFA application.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            kwargs = {
                "UserPoolId": self.user_pool_id,
                "ClientId": self.client_id,
```

```
                "ChallengeName": "SOFTWARE_TOKEN_MFA",
                "Session": session,
                "ChallengeResponses": {
                    "USERNAME": user_name,
                    "SOFTWARE_TOKEN_MFA_CODE": mfa_code,
                },
            }
            if self.client_secret is not None:
                kwargs["ChallengeResponses"]["SECRET_HASH"] = self._secret_hash(
                    user_name
                )
            response =
self.cognito_idp_client.admin_respond_to_auth_challenge(**kwargs)
            auth_result = response["AuthenticationResult"]
        except ClientError as err:
            if err.response["Error"]["Code"] == "ExpiredCodeException":
                logger.warning(
                    "Your MFA code has expired or has been used already. You
might have "
                    "to wait a few seconds until your app shows you a new code."
                )
            else:
                logger.error(
                    "Couldn't respond to mfa challenge for %s. Here's why: %s:
%s",
                    user_name,
                    err.response["Error"]["Code"],
                    err.response["Error"]["Message"],
                )
                raise
        else:
            return auth_result


    def confirm_mfa_device(
        self,
        user_name,
        device_key,
        device_group_key,
        device_password,
        access_token,
        aws_srp,
    ):
        """
```

```
        Confirms an MFA device to be tracked by Amazon Cognito. When a device is
        tracked, its key and password can be used to sign in without requiring a
new
        MFA code from the MFA application.

        :param user_name: The user that is associated with the device.
        :param device_key: The key of the device, returned by Amazon Cognito.
        :param device_group_key: The group key of the device, returned by Amazon
Cognito.
        :param device_password: The password that is associated with the device.
        :param access_token: The user's access token.
        :param aws_srp: A class that helps with Secure Remote Password (SRP)
                        calculations. The scenario associated with this example
uses
                        the warrant package.
        :return: True when the user must confirm the device. Otherwise, False.
When
                 False, the device is automatically confirmed and tracked.
        """
        srp_helper = aws_srp.AWSSRP(
            username=user_name,
            password=device_password,
            pool_id="_",
            client_id=self.client_id,
            client_secret=None,
            client=self.cognito_idp_client,
        )
        device_and_pw = f"{device_group_key}{device_key}:{device_password}"
        device_and_pw_hash = aws_srp.hash_sha256(device_and_pw.encode("utf-8"))
        salt = aws_srp.pad_hex(aws_srp.get_random(16))
        x_value = aws_srp.hex_to_long(aws_srp.hex_hash(salt +
device_and_pw_hash))
        verifier = aws_srp.pad_hex(pow(srp_helper.val_g, x_value,
srp_helper.big_n))
        device_secret_verifier_config = {
            "PasswordVerifier": base64.standard_b64encode(
                bytearray.fromhex(verifier)
            ).decode("utf-8"),
            "Salt":
base64.standard_b64encode(bytearray.fromhex(salt)).decode("utf-8"),
        }
        try:
            response = self.cognito_idp_client.confirm_device(
                AccessToken=access_token,
```

```
                DeviceKey=device_key,
                DeviceSecretVerifierConfig=device_secret_verifier_config,
            )
            user_confirm = response["UserConfirmationNecessary"]
        except ClientError as err:
            logger.error(
                "Couldn't confirm mfa device %s. Here's why: %s: %s",
                device_key,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return user_confirm


    def sign_in_with_tracked_device(
        self,
        user_name,
        password,
        device_key,
        device_group_key,
        device_password,
        aws_srp,
    ):
        """
        Signs in to Amazon Cognito as a user who has a tracked device. Signing in
        with a tracked device lets a user sign in without entering a new MFA
 code.

        Signing in with a tracked device requires that the client respond to the
 SRP
        protocol. The scenario associated with this example uses the warrant
 package
        to help with SRP calculations.

        For more information on SRP, see https://en.wikipedia.org/wiki/
Secure_Remote_Password_protocol.

        :param user_name: The user that is associated with the device.
        :param password: The user's password.
        :param device_key: The key of a tracked device.
        :param device_group_key: The group key of a tracked device.
        :param device_password: The password that is associated with the device.
```

```
        :param aws_srp: A class that helps with SRP calculations. The scenario
                        associated with this example uses the warrant package.
        :return: The result of the authentication. When successful, this contains
an
                 access token for the user.
        """
        try:
            srp_helper = aws_srp.AWSSRP(
                username=user_name,
                password=device_password,
                pool_id="_",
                client_id=self.client_id,
                client_secret=None,
                client=self.cognito_idp_client,
            )

            response_init = self.cognito_idp_client.initiate_auth(
                ClientId=self.client_id,
                AuthFlow="USER_PASSWORD_AUTH",
                AuthParameters={
                    "USERNAME": user_name,
                    "PASSWORD": password,
                    "DEVICE_KEY": device_key,
                },
            )
            if response_init["ChallengeName"] != "DEVICE_SRP_AUTH":
                raise RuntimeError(
                    f"Expected DEVICE_SRP_AUTH challenge but got
{response_init['ChallengeName']}."
                )

            auth_params = srp_helper.get_auth_params()
            auth_params["DEVICE_KEY"] = device_key
            response_auth = self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_SRP_AUTH",
                ChallengeResponses=auth_params,
            )
            if response_auth["ChallengeName"] != "DEVICE_PASSWORD_VERIFIER":
                raise RuntimeError(
                    f"Expected DEVICE_PASSWORD_VERIFIER challenge but got "
                    f"{response_init['ChallengeName']}."
                )
```

```
            challenge_params = response_auth["ChallengeParameters"]
            challenge_params["USER_ID_FOR_SRP"] = device_group_key + device_key
            cr = srp_helper.process_challenge(challenge_params, {"USERNAME":
 user_name})
            cr["USERNAME"] = user_name
            cr["DEVICE_KEY"] = device_key
            response_verifier =
 self.cognito_idp_client.respond_to_auth_challenge(
                ClientId=self.client_id,
                ChallengeName="DEVICE_PASSWORD_VERIFIER",
                ChallengeResponses=cr,
            )
            auth_tokens = response_verifier["AuthenticationResult"]
        except ClientError as err:
            logger.error(
                "Couldn't start client sign in for %s. Here's why: %s: %s",
                user_name,
                err.response["Error"]["Code"],
                err.response["Error"]["Message"],
            )
            raise
        else:
            return auth_tokens
```

创建运行场景的类。此示例还注册了一个 MFA 设备以通过 Amazon Cognito 进行跟踪，并向您演示如何使用来自被跟踪设备的密码和信息进行登录。这样，就无需输入新的 MFA 代码。

```
def run_scenario(cognito_idp_client, user_pool_id, client_id):
    logging.basicConfig(level=logging.INFO, format="%(levelname)s: %(message)s")

    print("-" * 88)
    print("Welcome to the Amazon Cognito user signup with MFA demo.")
    print("-" * 88)

    cog_wrapper = CognitoIdentityProviderWrapper(
        cognito_idp_client, user_pool_id, client_id
    )
```

```
    user_name = q.ask("Let's sign up a new user. Enter a user name: ",
q.non_empty)
    password = q.ask("Enter a password for the user: ", q.non_empty)
    email = q.ask("Enter a valid email address that you own: ", q.non_empty)
    confirmed = cog_wrapper.sign_up_user(user_name, password, email)
    while not confirmed:
        print(
            f"User {user_name} requires confirmation. Check {email} for "
            f"a verification code."
        )
        confirmation_code = q.ask("Enter the confirmation code from the email: ")
        if not confirmation_code:
            if q.ask("Do you need another confirmation code (y/n)? ",
q.is_yesno):
                delivery = cog_wrapper.resend_confirmation(user_name)
                print(
                    f"Confirmation code sent by {delivery['DeliveryMedium']} "
                    f"to {delivery['Destination']}."
                )
        else:
            confirmed = cog_wrapper.confirm_user_sign_up(user_name,
confirmation_code)
    print(f"User {user_name} is confirmed and ready to use.")
    print("-" * 88)

    print("Let's get a list of users in the user pool.")
    q.ask("Press Enter when you're ready.")
    users = cog_wrapper.list_users()
    if users:
        print(f"Found {len(users)} users:")
        pp(users)
    else:
        print("No users found.")
    print("-" * 88)

    print("Let's sign in and get an access token.")
    auth_tokens = None
    challenge = "ADMIN_USER_PASSWORD_AUTH"
    response = {}
    while challenge is not None:
        if challenge == "ADMIN_USER_PASSWORD_AUTH":
            response = cog_wrapper.start_sign_in(user_name, password)
            challenge = response["ChallengeName"]
        elif response["ChallengeName"] == "MFA_SETUP":
```

```
            print("First, we need to set up an MFA application.")
            qr_img = qrcode.make(
                f"otpauth://totp/{user_name}?secret={response['SecretCode']}"
            )
            qr_img.save("qr.png")
            q.ask(
                "Press Enter to see a QR code on your screen. Scan it into an MFA
"
                "application, such as Google Authenticator."
            )
            webbrowser.open("qr.png")
            mfa_code = q.ask(
                "Enter the verification code from your MFA application: ",
q.non_empty
            )
            response = cog_wrapper.verify_mfa(response["Session"], mfa_code)
            print(f"MFA device setup {response['Status']}")
            print("Now that an MFA application is set up, let's sign in again.")
            print(
                "You might have to wait a few seconds for a new MFA code to
appear in "
                "your MFA application."
            )
            challenge = "ADMIN_USER_PASSWORD_AUTH"
        elif response["ChallengeName"] == "SOFTWARE_TOKEN_MFA":
            auth_tokens = None
            while auth_tokens is None:
                mfa_code = q.ask(
                    "Enter a verification code from your MFA application: ",
q.non_empty
                )
                auth_tokens = cog_wrapper.respond_to_mfa_challenge(
                    user_name, response["Session"], mfa_code
                )
            print(f"You're signed in as {user_name}.")
            print("Here's your access token:")
            pp(auth_tokens["AccessToken"])
            print("And your device information:")
            pp(auth_tokens["NewDeviceMetadata"])
            challenge = None
        else:
            raise Exception(f"Got unexpected challenge
{response['ChallengeName']}")
    print("-" * 88)
```

```
    device_group_key = auth_tokens["NewDeviceMetadata"]["DeviceGroupKey"]
    device_key = auth_tokens["NewDeviceMetadata"]["DeviceKey"]
    device_password = base64.standard_b64encode(os.urandom(40)).decode("utf-8")

    print("Let's confirm your MFA device so you don't have re-enter MFA tokens
 for it.")
    q.ask("Press Enter when you're ready.")
    cog_wrapper.confirm_mfa_device(
        user_name,
        device_key,
        device_group_key,
        device_password,
        auth_tokens["AccessToken"],
        aws_srp,
    )
    print(f"Your device {device_key} is confirmed.")
    print("-" * 88)

    print(
        f"Now let's sign in as {user_name} from your confirmed device
 {device_key}.\n"
        f"Because this device is tracked by Amazon Cognito, you won't have to re-
enter an MFA code."
    )
    q.ask("Press Enter when ready.")
    auth_tokens = cog_wrapper.sign_in_with_tracked_device(
        user_name, password, device_key, device_group_key, device_password,
 aws_srp
    )
    print("You're signed in. Your access token is:")
    pp(auth_tokens["AccessToken"])
    print("-" * 88)

    print("Don't forget to delete your user pool when you're done with this
 example.")
    print("\nThanks for watching!")
    print("-" * 88)


def main():
    parser = argparse.ArgumentParser(
        description="Shows how to sign up a new user with Amazon Cognito and
 associate "
```

```
        "the user with an MFA application for multi-factor authentication."
    )
    parser.add_argument(
        "user_pool_id", help="The ID of the user pool to use for the example."
    )
    parser.add_argument(
        "client_id", help="The ID of the client application to use for the
 example."
    )
    args = parser.parse_args()
    try:
        run_scenario(boto3.client("cognito-idp"), args.user_pool_id,
 args.client_id)
    except Exception:
        logging.exception("Something went wrong with the demo.")


if __name__ == "__main__":
    main()
```

- 有关 API 详细信息，请参阅《Amazon SDK for Python（Boto3）API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp
  - InitiateAuth
  - ListUsers
  - ResendConfirmationCode
  - RespondToAuthChallenge
  - SignUp
  - VerifySoftwareToken

Swift

适用于 Swift 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

`Package.swift` 文件。

```
// swift-tools-version: 5.9
//
// The swift-tools-version declares the minimum version of Swift required to
// build this package.

import PackageDescription

let package = Package(
    name: "http-config",
    // Let Xcode know the minimum Apple platforms supported.
    platforms: [
        .macOS(.v13),
        .iOS(.v15)
    ],
    dependencies: [
        // Dependencies declare other packages that this package depends on.
        .package(
            url: "https://github.com/awslabs/aws-sdk-swift",
            from: "1.0.0"),
        .package(
            url: "https://github.com/apple/swift-argument-parser.git",
            branch: "main"
        )
    ],
    targets: [
        // Targets are the basic building blocks of a package, defining a module
 or a test suite.
        // Targets can depend on other targets in this package and products
        // from dependencies.
        .executableTarget(
            name: "cognito-scenario",
```

```
            dependencies: [
                .product(name: "AWSS3", package: "aws-sdk-swift"),
                .product(name: "ArgumentParser", package: "swift-argument-
parser")
            ],
            path: "Sources")

    ]
)
```

Swift 代码文件。

```
// An example demonstrating various features of Amazon Cognito. Before running
// this Swift code example, set up your development environment, including
// your credentials.
//
// For more information, see the following documentation:
// https://docs.aws.amazon.com/sdk-for-kotlin/latest/developer-guide/setup.html
//
// TIP: To set up the required user pool, run the AWS Cloud Development Kit
// (AWS CDK) script provided in this GitHub repo at
// resources/cdk/cognito_scenario_user_pool_with_mfa.
//
// This example performs the following functions:
//
// 1. Invokes the signUp method to sign up a user.
// 2. Invokes the adminGetUser method to get the user's confirmation status.
// 3. Invokes the ResendConfirmationCode method if the user requested another
//    code.
// 4. Invokes the confirmSignUp method.
// 5. Invokes the initiateAuth to sign in. This results in being prompted to
//    set up TOTP (time-based one-time password). (The response is
//    “ChallengeName”: “MFA_SETUP”).
// 6. Invokes the AssociateSoftwareToken method to generate a TOTP MFA private
//    key. This can be used with Google Authenticator.
// 7. Invokes the VerifySoftwareToken method to verify the TOTP and register
//    for MFA.
// 8. Invokes the AdminInitiateAuth to sign in again. This results in being
//    prompted to submit a TOTP (Response: “ChallengeName”:
//    “SOFTWARE_TOKEN_MFA”).
// 9. Invokes the AdminRespondToAuthChallenge to get back a token.
```

```
import ArgumentParser
import AWSClientRuntime
import Foundation

import AWSCognitoIdentityProvider

struct ExampleCommand: ParsableCommand {
    @Argument(help: "The application clientId.")
    var clientId: String
    @Argument(help: "The user pool ID to use.")
    var poolId: String
    @Option(help: "Name of the Amazon Region to use")
    var region = "us-east-1"

    static var configuration = CommandConfiguration(
        commandName: "cognito-scenario",
        abstract: """
        Demonstrates various features of Amazon Cognito.
        """,
        discussion: """
        """
    )

    /// Prompt for an input string of at least a minimum length.
    ///
    /// - Parameters:
    ///   - prompt: The prompt string to display.
    ///   - minLength: The minimum number of characters to allow in the
    ///     response. Default value is 0.
    ///
    /// - Returns: The entered string.
    func stringRequest(_ prompt: String, minLength: Int = 1) -> String {
        while true {
            print(prompt, terminator: "")
            let str = readLine()

            guard let str else {
                continue
            }
            if str.count >= minLength {
                return str
            } else {
                print("*** Response must be at least \(minLength) character(s)
 long.")
```

```
            }
        }
    }

    /// Ask a yes/no question.
    ///
    /// - Parameter prompt: A prompt string to print.
    ///
    /// - Returns: `true` if the user answered "Y", otherwise `false`.
    func yesNoRequest(_ prompt: String) -> Bool {
        while true {
            let answer = stringRequest(prompt).lowercased()
            if answer == "y" || answer == "n" {
                return answer == "y"
            }
        }
    }

    /// Get information about a specific user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The Amazon Cognito Identity Provider client to use.
    ///   - userName: The user to retrieve information about.
    ///   - userPoolId: The user pool to search for the specified user.
    ///
    /// - Returns: `true` if the user's information was successfully
    ///   retrieved. Otherwise returns `false`.
    func adminGetUser(cipClient: CognitoIdentityProviderClient, userName: String,
                      userPoolId: String) async -> Bool {
        do {
            let output = try await cipClient.adminGetUser(
                input: AdminGetUserInput(
                    userPoolId: userPoolId,
                    username: userName
                )
            )

            guard let userStatus = output.userStatus else {
                print("*** Unable to get the user's status.")
                return false
            }

            print("User status: \(userStatus)")
            return true
```

```
        } catch {
            return false
        }
    }

    /// Create a new user in a user pool.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The ID of the app client to create a user for.
    ///   - userName: The username for the new user.
    ///   - password: The new user's password.
    ///   - email: The new user's email address.
    ///
    /// - Returns: `true` if successful; otherwise `false`.
    func signUp(cipClient: CognitoIdentityProviderClient, clientId: String,
userName: String, password: String, email: String) async -> Bool {
        let emailAttr = CognitoIdentityProviderClientTypes.AttributeType(
            name: "email",
            value: email
        )

        let userAttrsList = [emailAttr]

        do {
            _ = try await cipClient.signUp(
                input: SignUpInput(
                    clientId: clientId,
                    password: password,
                    userAttributes: userAttrsList,
                    username: userName
                )

            )

            print("=====> User \(userName) signed up.")
        } catch _ as AWSCognitoIdentityProvider.UsernameExistsException {
            print("*** The username \(userName) already exists. Please use a
different one.")
            return false
        } catch let error as AWSCognitoIdentityProvider.InvalidPasswordException
{
            print("*** Error: The specified password is invalid. Reason:
\(error.properties.message ?? "<none available>").")
```

```
            return false
        } catch _ as AWSCognitoIdentityProvider.ResourceNotFoundException {
            print("*** Error: The specified client ID (\(clientId)) doesn't
exist.")
            return false
        } catch {
            print("*** Unexpected error: \(error)")
            return false
        }

        return true
    }

    /// Requests a new confirmation code be sent to the given user's contact
    /// method.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The application client ID.
    ///   - userName: The user to resend a code for.
    ///
    /// - Returns: `true` if a new code was sent successfully, otherwise
    ///   `false`.
    func resendConfirmationCode(cipClient: CognitoIdentityProviderClient,
clientId: String,
                                userName: String) async -> Bool {
        do {
            let output = try await cipClient.resendConfirmationCode(
                input: ResendConfirmationCodeInput(
                    clientId: clientId,
                    username: userName
                )
            )

            guard let deliveryMedium = output.codeDeliveryDetails?.deliveryMedium
else {
                print("*** Unable to get the delivery method for the resent
code.")
                return false
            }

            print("=====> A new code has been sent by \(deliveryMedium)")
            return true
        } catch {
```

```
            print("*** Unable to resend the confirmation code to user
\(userName).")
            return false
        }
    }

    /// Submit a confirmation code for the specified user. This is the code as
    /// entered by the user after they've received it by email or text
    /// message.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID the user is signing up for.
    ///   - userName: The username of the user whose code is being sent.
    ///   - code: The user's confirmation code.
    ///
    /// - Returns: `true` if the code was successfully confirmed; otherwise
`false`.
    func confirmSignUp(cipClient: CognitoIdentityProviderClient, clientId:
String,
                       userName: String, code: String) async -> Bool {
        do {
            _ = try await cipClient.confirmSignUp(
                input: ConfirmSignUpInput(
                    clientId: clientId,
                    confirmationCode: code,
                    username: userName
                )
            )

            print("=====> \(userName) has been confirmed.")
            return true
        } catch {
            print("=====> \(userName)'s code was entered incorrectly.")
            return false
        }
    }

    /// Begin an authentication session.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongitoIdentityProviderClient` to use.
    ///   - clientId: The app client ID to use.
    ///   - userName: The username to check.
```

```
    ///   - password: The user's password.
    ///   - userPoolId: The user pool to use.
    ///
    /// - Returns: The session token associated with this authentication
    ///   session.
    func initiateAuth(cipClient: CognitoIdentityProviderClient, clientId: String,
                      userName: String, password: String,
                      userPoolId: String) async -> String? {
        var authParams: [String: String] = [:]

        authParams["USERNAME"] = userName
        authParams["PASSWORD"] = password

        do {
            let output = try await cipClient.adminInitiateAuth(
                input: AdminInitiateAuthInput(
                    authFlow:
CognitoIdentityProviderClientTypes.AuthFlowType.adminUserPasswordAuth,
                    authParameters: authParams,
                    clientId: clientId,
                    userPoolId: userPoolId
                )
            )

            guard let challengeName = output.challengeName else {
                print("*** Invalid response from the auth service.")
                return nil
            }

            print("=====> Response challenge is \(challengeName)")

            return output.session
        } catch _ as UserNotFoundException {
            print("*** The specified username, \(userName), doesn't exist.")
            return nil
        } catch _ as UserNotConfirmedException {
            print("*** The user \(userName) has not been confirmed.")
            return nil
        } catch {
            print("*** An unexpected error occurred.")
            return nil
        }
    }
```

```
    /// Request and display an MFA secret token that the user should enter
    /// into their authenticator to set it up for the user account.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - authSession: The authentication session to request an MFA secret
    ///     for.
    ///
    /// - Returns: A string containing the MFA secret token that should be
    ///   entered into the authenticator software.
    func getSecretForAppMFA(cipClient: CognitoIdentityProviderClient,
authSession: String?) async -> String? {
        do {
            let output = try await cipClient.associateSoftwareToken(
                input: AssociateSoftwareTokenInput(
                    session: authSession
                )
            )

            guard let secretCode = output.secretCode else {
                print("*** Unable to get the secret code")
                return nil
            }

            print("=====> Enter this token into Google Authenticator:
\(secretCode)")
            return output.session
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return nil
        } catch {
            print("*** An unexpected error occurred getting the secret for the
app's MFA.")
            return nil
        }
    }

    /// Confirm that the user's TOTP authenticator is configured correctly by
    /// sending a code to it to check that it matches successfully.
    ///
    /// - Parameters:
    ///   - cipClient: The `CongnitoIdentityProviderClient` to use.
    ///   - session: An authentication session previously returned by an
    ///     `associateSoftwareToken()` call.
```

```
    ///   - mfaCode: The 6-digit code currently displayed by the user's
    ///     authenticator, as provided by the user.
    func verifyTOTP(cipClient: CognitoIdentityProviderClient, session: String?,
mfaCode: String?) async {
        do {
            let output = try await cipClient.verifySoftwareToken(
                input: VerifySoftwareTokenInput(
                    session: session,
                    userCode: mfaCode
                )
            )

            guard let tokenStatus = output.status else {
                print("*** Unable to get the token's status.")
                return
            }
            print("=====> The token's status is: \(tokenStatus)")
        } catch _ as SoftwareTokenMFANotFoundException {
            print("*** The specified user pool isn't configured for MFA.")
            return
        } catch _ as CodeMismatchException {
            print("*** The specified MFA code doesn't match the expected value.")
            return
        } catch _ as UserNotFoundException {
            print("*** The specified username doesn't exist.")
            return
        } catch _ as UserNotConfirmedException {
            print("*** The user has not been confirmed.")
            return
        } catch {
            print("*** Error verifying the MFA token!")
            return
        }
    }

    /// Respond to the authentication challenge received from Cognito after
    /// initiating an authentication session. This involves sending a current
    /// MFA code to the service.
    ///
    /// - Parameters:
    ///   - cipClient: The `CognitoIdentityProviderClient` to use.
    ///   - userName: The user's username.
    ///   - clientId: The app client ID.
    ///   - userPoolId: The user pool to sign into.
```

```
   ///   - mfaCode: The 6-digit MFA code currently displayed by the user's
   ///     authenticator.
   ///   - session: The authentication session to continue processing.
   func adminRespondToAuthChallenge(cipClient: CognitoIdentityProviderClient,
userName: String,
                                    clientId: String, userPoolId: String,
mfaCode: String,
                                    session: String) async {
       print("=====> SOFTWARE_TOKEN_MFA challenge is generated...")

       var challengeResponsesOb: [String: String] = [:]
       challengeResponsesOb["USERNAME"] = userName
       challengeResponsesOb["SOFTWARE_TOKEN_MFA_CODE"] = mfaCode

       do {
           let output = try await cipClient.adminRespondToAuthChallenge(
               input: AdminRespondToAuthChallengeInput(
                   challengeName:
CognitoIdentityProviderClientTypes.ChallengeNameType.softwareTokenMfa,
                   challengeResponses: challengeResponsesOb,
                   clientId: clientId,
                   session: session,
                   userPoolId: userPoolId
               )
           )

           guard let authenticationResult = output.authenticationResult else {
               print("*** Unable to get authentication result.")
               return
           }

           print("=====> Authentication result (JWTs are redacted):")
           print(authenticationResult)
       } catch _ as SoftwareTokenMFANotFoundException {
           print("*** The specified user pool isn't configured for MFA.")
           return
       } catch _ as CodeMismatchException {
           print("*** The specified MFA code doesn't match the expected value.")
           return
       } catch _ as UserNotFoundException {
           print("*** The specified username, \(userName), doesn't exist.")
           return
       } catch _ as UserNotConfirmedException {
           print("*** The user \(userName) has not been confirmed.")
```

```
            return
        } catch let error as NotAuthorizedException {
            print("*** Unauthorized access. Reason: \(error.properties.message ??
"<unknown>")")
        } catch {
            print("*** Error responding to the MFA challenge.")
            return
        }
    }

    /// Called by ``main()`` to run the bulk of the example.
    func runAsync() async throws {
        let config = try await
CognitoIdentityProviderClient.CognitoIdentityProviderClientConfiguration(region:
region)
        let cipClient = CognitoIdentityProviderClient(config: config)

        print("""
              This example collects information about a user, then creates that
user in the
              specified user pool. Then, it enables Multi-Factor Authentication
(MFA) for that
              user by associating an authenticator application (such as Google
Authenticator
              or a password manager that supports TOTP). Then, the user uses a
code from their
              authenticator application to sign in.

              """)

        let userName = stringRequest("Please enter a new username: ")
        let password = stringRequest("Enter a password: ")
        let email = stringRequest("Enter your email address: ", minLength: 5)

        // Submit the sign-up request to AWS.

        print("==> Signing up user \(userName)...")
        if await signUp(cipClient: cipClient, clientId: clientId,
                        userName: userName, password: password,
                        email: email) == false {
            return
        }

        // Check the user's status. This time, it should come back "unconfirmed".
```

```
        print("==> Getting the status of user \(userName) from the user pool
(should be 'unconfirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }

        // Ask the user if they want a replacement code sent, such as if the
        // code hasn't arrived yet. If the user responds with a "yes," send a
        // new code.

        if yesNoRequest("==> A confirmation code was sent to \(userName). Would
you like to send a new code (Y/N)? ") {
            print("==> Sending a new confirmation code...")
            if await resendConfirmationCode(cipClient: cipClient, clientId:
clientId, userName: userName) == false {
                return
            }
        }

        // Ask the user to enter the confirmation code, then send it to Amazon
        // Cognito to verify it.

        let code = stringRequest("==> Enter the confirmation code sent to
\(userName): ")
        if await confirmSignUp(cipClient: cipClient, clientId: clientId,
userName: userName, code: code) == false {
            // The code didn't match. Your application may wish to offer to
            // re-send the confirmation code here and try again.
            return
        }

        // Check the user's status again. This time it should come back
        // "confirmed".

        print("==> Rechecking status of user \(userName) in the user pool (should
be 'confirmed')...")
        if await adminGetUser(cipClient: cipClient, userName: userName,
userPoolId: poolId) == false {
            return
        }
        // Check the challenge mode. Here, it should be "mfaSetup", indicating
        // that the user needs to add MFA before using it. This returns a
```

```
        // session that can be used to register MFA, or nil if an error occurs.

        let authSession = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                             userName: userName, password:
password,
                                             userPoolId: poolId)
        if authSession == nil {
            return
        }

        // Ask Cognito for an MFA secret token that the user should enter into
        // their authenticator software (such as Google Authenticator) or
        // password manager to configure it for this user account. This
        // returns a new session that should be used for the new stage of the
        // authentication process.

        let newSession = await getSecretForAppMFA(cipClient: cipClient,
authSession: authSession)
        if newSession == nil {
            return
        }

        // Ask the user to enter the current 6-digit code displayed by their
        // authenticator. Then verify that it matches the value expected for
        // the session.

        let mfaCode1 = stringRequest("==> Enter the 6-digit code displayed in
your authenticator: ",
                                     minLength: 6)
        await verifyTOTP(cipClient: cipClient, session: newSession, mfaCode:
mfaCode1)

        // Ask the user to authenticate now that the authenticator has been
        // configured. This creates a new session using the user's username
        // and password as already entered.

        print("\nNow starting the sign-in process for user \(userName)...\n")

        let session2 = await initiateAuth(cipClient: cipClient, clientId:
clientId,
                                          userName: userName, password: password,
userPoolId: poolId)
        guard let session2 else {
```

```
            return
        }

        // Now that we have a new auth session, `session2`, ask the user for a
        // new 6-digit code from their authenticator, and send it to the auth
        // session.

        let mfaCode2 = stringRequest("==> Wait for your authenticator to show a
 new 6-digit code, then enter it: ",
                                     minLength: 6)
        await adminRespondToAuthChallenge(cipClient: cipClient, userName:
 userName,
                                          clientId: clientId, userPoolId: poolId,
                                          mfaCode: mfaCode2, session: session2)
    }
}

/// The program's asynchronous entry point.
@main
struct Main {
    static func main() async {
        let args = Array(CommandLine.arguments.dropFirst())

        do {
            let command = try ExampleCommand.parse(args)
            try await command.runAsync()
        } catch {
            ExampleCommand.exit(withError: error)
        }
    }
}
```

- 有关 API 详细信息，请参阅《Amazon SDK for Swift API 参考》中的以下主题。
  - AdminGetUser
  - AdminInitiateAuth
  - AdminRespondToAuthChallenge
  - AssociateSoftwareToken
  - ConfirmDevice
  - ConfirmSignUp

- InitiateAuth
- ListUsers
- ResendConfirmationCode
- RespondToAuthChallenge
- SignUp
- VerifySoftwareToken

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用软件开发工具包在 Amazon Cognito 用户身份验证后，使用 Lambda 函数编写自定义活动数据 Amazon

以下代码示例显示了在完成 Amazon Cognito 用户身份验证后如何使用 Lambda 函数写入自定义活动数据。

- 使用管理员功能将用户添加到用户池。
- 配置用户池以调用 `PostAuthentication` 触发器的 Lambda 函数。
- 将新用户登录到 Amazon Cognito 控制台。
- Lambda 函数将自定义信息写入 CloudWatch 日志和 DynamoDB 表。
- 从 DynamoDB 表获取并显示自定义数据，然后清理资源。

Go

适用于 Go V2 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。在 Amazon 代码示例存储库中查找完整示例，了解如何进行设置和运行。

在命令提示符中运行交互式场景。

```
import (
```

```
 "context"
 "errors"
 "log"
 "strings"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// ActivityLog separates the steps of this scenario into individual functions so
 that
// they are simpler to read and understand.
type ActivityLog struct {
 helper       IScenarioHelper
 questioner   demotools.IQuestioner
 resources    Resources
 cognitoActor *actions.CognitoActions
}

// NewActivityLog constructs a new activity log runner.
func NewActivityLog(sdkConfig aws.Config, questioner demotools.IQuestioner,
 helper IScenarioHelper) ActivityLog {
 scenario := ActivityLog{
  helper:       helper,
  questioner:   questioner,
  resources:    Resources{},
  cognitoActor: &actions.CognitoActions{CognitoClient:
 cognitoidentityprovider.NewFromConfig(sdkConfig)},
 }
 scenario.resources.init(scenario.cognitoActor, questioner)
 return scenario
}

// AddUserToPool selects a user from the known users table and uses administrator
 credentials to add the user to the user pool.
func (runner *ActivityLog) AddUserToPool(ctx context.Context, userPoolId string,
 tableName string) (string, string) {
 log.Println("To facilitate this example, let's add a user to the user pool using
 administrator privileges.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
```

```
  panic(err)
 }
 user := users.Users[0]
 log.Printf("Adding known user %v to the user pool.\n", user.UserName)
 err = runner.cognitoActor.AdminCreateUser(ctx, userPoolId, user.UserName,
 user.UserEmail)
 if err != nil {
  panic(err)
 }
 pwSet := false
 password := runner.questioner.AskPassword("\nEnter a password that has at least
 eight characters, uppercase, lowercase, numbers and symbols.\n"+
  "(the password will not display as you type):", 8)
 for !pwSet {
  log.Printf("\nSetting password for user '%v'.\n", user.UserName)
  err = runner.cognitoActor.AdminSetUserPassword(ctx, userPoolId, user.UserName,
 password)
  if err != nil {
   var invalidPassword *types.InvalidPasswordException
   if errors.As(err, &invalidPassword) {
    password = runner.questioner.AskPassword("\nEnter another password:", 8)
   } else {
    panic(err)
   }
  } else {
   pwSet = true
  }
 }

 log.Println(strings.Repeat("-", 88))

 return user.UserName, password
}

// AddActivityLogTrigger adds a Lambda handler as an invocation target for the
 PostAuthentication trigger.
func (runner *ActivityLog) AddActivityLogTrigger(ctx context.Context, userPoolId
 string, activityLogArn string) {
 log.Println("Let's add a Lambda function to handle the PostAuthentication
 trigger from Cognito.\n" +
  "This trigger happens after a user is authenticated, and lets your function
 take action, such as logging\n" +
  "the outcome.")
 err := runner.cognitoActor.UpdateTriggers(
```

```
  ctx, userPoolId,
  actions.TriggerInfo{Trigger: actions.PostAuthentication, HandlerArn:
 aws.String(activityLogArn)})
 if err != nil {
  panic(err)
 }
 runner.resources.triggers = append(runner.resources.triggers,
 actions.PostAuthentication)
 log.Printf("Lambda function %v added to user pool %v to handle
 PostAuthentication Cognito trigger.\n",
  activityLogArn, userPoolId)

 log.Println(strings.Repeat("-", 88))
}

// SignInUser signs in as the specified user.
func (runner *ActivityLog) SignInUser(ctx context.Context, clientId string,
 userName string, password string) {
 log.Printf("Now we'll sign in user %v and check the results in the logs and the
 DynamoDB table.", userName)
 runner.questioner.Ask("Press Enter when you're ready.")
 authResult, err := runner.cognitoActor.SignIn(ctx, clientId, userName, password)
 if err != nil {
  panic(err)
 }
 log.Println("Sign in successful.",
  "The PostAuthentication Lambda handler writes custom information to CloudWatch
 Logs.")

 runner.resources.userAccessTokens = append(runner.resources.userAccessTokens,
 *authResult.AccessToken)
}

// GetKnownUserLastLogin gets the login info for a user from the Amazon DynamoDB
 table and displays it.
func (runner *ActivityLog) GetKnownUserLastLogin(ctx context.Context, tableName
 string, userName string) {
 log.Println("The PostAuthentication handler also writes login data to the
 DynamoDB table.")
 runner.questioner.Ask("Press Enter when you're ready to continue.")
 users, err := runner.helper.GetKnownUsers(ctx, tableName)
 if err != nil {
  panic(err)
 }
```

```
 for _, user := range users.Users {
  if user.UserName == userName {
   log.Println("The last login info for the user in the known users table is:")
   log.Printf("\t%+v", *user.LastLogin)
  }
 }
 log.Println(strings.Repeat("-", 88))
}

// Run runs the scenario.
func (runner *ActivityLog) Run(ctx context.Context, stackName string) {
 defer func() {
  if r := recover(); r != nil {
   log.Println("Something went wrong with the demo.")
   runner.resources.Cleanup(ctx)
  }
 }()

 log.Println(strings.Repeat("-", 88))
 log.Printf("Welcome\n")

 log.Println(strings.Repeat("-", 88))

 stackOutputs, err := runner.helper.GetStackOutputs(ctx, stackName)
 if err != nil {
  panic(err)
 }
 runner.resources.userPoolId = stackOutputs["UserPoolId"]
 runner.helper.PopulateUserTable(ctx, stackOutputs["TableName"])
 userName, password := runner.AddUserToPool(ctx, stackOutputs["UserPoolId"],
 stackOutputs["TableName"])

 runner.AddActivityLogTrigger(ctx, stackOutputs["UserPoolId"],
 stackOutputs["ActivityLogFunctionArn"])
 runner.SignInUser(ctx, stackOutputs["UserPoolClientId"], userName, password)
 runner.helper.ListRecentLogEvents(ctx, stackOutputs["ActivityLogFunction"])
 runner.GetKnownUserLastLogin(ctx, stackOutputs["TableName"], userName)

 runner.resources.Cleanup(ctx)

 log.Println(strings.Repeat("-", 88))
 log.Println("Thanks for watching!")
 log.Println(strings.Repeat("-", 88))
}
```

使用 Lambda 函数处理 PostAuthentication 触发器。

```
import (
 "context"
 "fmt"
 "log"
 "os"
 "time"

 "github.com/aws/aws-lambda-go/events"
 "github.com/aws/aws-lambda-go/lambda"
 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/config"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 dynamodbtypes "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

const TABLE_NAME = "TABLE_NAME"

// LoginInfo defines structured login data that can be marshalled to a DynamoDB
 format.
type LoginInfo struct {
 UserPoolId string `dynamodbav:"UserPoolId"`
 ClientId   string `dynamodbav:"ClientId"`
 Time       string `dynamodbav:"Time"`
}

// UserInfo defines structured user data that can be marshalled to a DynamoDB
 format.
type UserInfo struct {
 UserName  string    `dynamodbav:"UserName"`
 UserEmail string    `dynamodbav:"UserEmail"`
 LastLogin LoginInfo `dynamodbav:"LastLogin"`
}

// GetKey marshals the user email value to a DynamoDB key format.
func (user UserInfo) GetKey() map[string]dynamodbtypes.AttributeValue {
 userEmail, err := attributevalue.Marshal(user.UserEmail)
```

```
 if err != nil {
  panic(err)
 }
 return map[string]dynamodbtypes.AttributeValue{"UserEmail": userEmail}
}

type handler struct {
 dynamoClient *dynamodb.Client
}

// HandleRequest handles the PostAuthentication event by writing custom data to
 the logs and
// to an Amazon DynamoDB table.
func (h *handler) HandleRequest(ctx context.Context,
 event events.CognitoEventUserPoolsPostAuthentication)
 (events.CognitoEventUserPoolsPostAuthentication, error) {
 log.Printf("Received post authentication trigger from %v for user '%v'",
 event.TriggerSource, event.UserName)
 tableName := os.Getenv(TABLE_NAME)
 user := UserInfo{
  UserName:  event.UserName,
  UserEmail: event.Request.UserAttributes["email"],
  LastLogin: LoginInfo{
   UserPoolId: event.UserPoolID,
   ClientId:   event.CallerContext.ClientID,
   Time:       time.Now().Format(time.UnixDate),
  },
 }
 // Write to CloudWatch Logs.
 fmt.Printf("%#v", user)

 // Also write to an external system. This examples uses DynamoDB to demonstrate.
 userMap, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshal to DynamoDB map. Here's why: %v\n", err)
 } else if len(userMap) == 0 {
  log.Printf("User info marshaled to an empty map.")
 } else {
  _, err := h.dynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
   Item:      userMap,
   TableName: aws.String(tableName),
  })
  if err != nil {
   log.Printf("Couldn't write to DynamoDB. Here's why: %v\n", err)
```

```
  } else {
   log.Printf("Wrote user info to DynamoDB table %v.\n", tableName)
  }
 }

 return event, nil
}

func main() {
 ctx := context.Background()
 sdkConfig, err := config.LoadDefaultConfig(ctx)
 if err != nil {
  log.Panicln(err)
 }
 h := handler{
  dynamoClient: dynamodb.NewFromConfig(sdkConfig),
 }
 lambda.Start(h.HandleRequest)
}
```

创建一个执行常见任务的结构。

```
import (
 "context"
 "log"
 "strings"
 "time"
 "user_pools_and_lambda_triggers/actions"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// IScenarioHelper defines common functions used by the workflows in this
 example.
type IScenarioHelper interface {
 Pause(secs int)
```

```
 GetStackOutputs(ctx context.Context, stackName string) (actions.StackOutputs,
 error)
 PopulateUserTable(ctx context.Context, tableName string)
 GetKnownUsers(ctx context.Context, tableName string) (actions.UserList, error)
 AddKnownUser(ctx context.Context, tableName string, user actions.User)
 ListRecentLogEvents(ctx context.Context, functionName string)
}

// ScenarioHelper contains AWS wrapper structs used by the workflows in this
 example.
type ScenarioHelper struct {
 questioner  demotools.IQuestioner
 dynamoActor *actions.DynamoActions
 cfnActor    *actions.CloudFormationActions
 cwlActor    *actions.CloudWatchLogsActions
 isTestRun   bool
}

// NewScenarioHelper constructs a new scenario helper.
func NewScenarioHelper(sdkConfig aws.Config, questioner demotools.IQuestioner)
 ScenarioHelper {
 scenario := ScenarioHelper{
  questioner:  questioner,
  dynamoActor: &actions.DynamoActions{DynamoClient:
 dynamodb.NewFromConfig(sdkConfig)},
  cfnActor:    &actions.CloudFormationActions{CfnClient:
 cloudformation.NewFromConfig(sdkConfig)},
  cwlActor:    &actions.CloudWatchLogsActions{CwlClient:
 cloudwatchlogs.NewFromConfig(sdkConfig)},
 }
 return scenario
}

// Pause waits for the specified number of seconds.
func (helper ScenarioHelper) Pause(secs int) {
 if !helper.isTestRun {
  time.Sleep(time.Duration(secs) * time.Second)
 }
}

// GetStackOutputs gets the outputs from the specified CloudFormation stack in a
 structured format.
func (helper ScenarioHelper) GetStackOutputs(ctx context.Context, stackName
 string) (actions.StackOutputs, error) {
```

```
 return helper.cfnActor.GetOutputs(ctx, stackName), nil
}

// PopulateUserTable fills the known user table with example data.
func (helper ScenarioHelper) PopulateUserTable(ctx context.Context, tableName
 string) {
 log.Printf("First, let's add some users to the DynamoDB %v table we'll use for
 this example.\n", tableName)
 err := helper.dynamoActor.PopulateTable(ctx, tableName)
 if err != nil {
  panic(err)
 }
}

// GetKnownUsers gets the users from the known users table in a structured
 format.
func (helper ScenarioHelper) GetKnownUsers(ctx context.Context, tableName string)
 (actions.UserList, error) {
 knownUsers, err := helper.dynamoActor.Scan(ctx, tableName)
 if err != nil {
  log.Printf("Couldn't get known users from table %v. Here's why: %v\n",
 tableName, err)
 }
 return knownUsers, err
}

// AddKnownUser adds a user to the known users table.
func (helper ScenarioHelper) AddKnownUser(ctx context.Context, tableName string,
 user actions.User) {
 log.Printf("Adding user '%v' with email '%v' to the DynamoDB known users
 table...\n",
  user.UserName, user.UserEmail)
 err := helper.dynamoActor.AddUser(ctx, tableName, user)
 if err != nil {
  panic(err)
 }
}

// ListRecentLogEvents gets the most recent log stream and events for the
 specified Lambda function and displays them.
func (helper ScenarioHelper) ListRecentLogEvents(ctx context.Context,
 functionName string) {
 log.Println("Waiting a few seconds to let Lambda write to CloudWatch Logs...")
 helper.Pause(10)
```

```
 log.Println("Okay, let's check the logs to find what's happened recently with
 your Lambda function.")
 logStream, err := helper.cwlActor.GetLatestLogStream(ctx, functionName)
 if err != nil {
  panic(err)
 }
 log.Printf("Getting some recent events from log stream %v\n",
 *logStream.LogStreamName)
 events, err := helper.cwlActor.GetLogEvents(ctx, functionName,
 *logStream.LogStreamName, 10)
 if err != nil {
  panic(err)
 }
 for _, event := range events {
  log.Printf("\t%v", *event.Message)
 }
 log.Println(strings.Repeat("-", 88))
}
```

创建一个封装 Amazon Cognito 操作的结构。

```
import (
 "context"
 "errors"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider"
 "github.com/aws/aws-sdk-go-v2/service/cognitoidentityprovider/types"
)

type CognitoActions struct {
 CognitoClient *cognitoidentityprovider.Client
}



// Trigger and TriggerInfo define typed data for updating an Amazon Cognito
 trigger.
type Trigger int
```

```
const (
 PreSignUp Trigger = iota
 UserMigration
 PostAuthentication
)

type TriggerInfo struct {
 Trigger    Trigger
 HandlerArn *string
}

// UpdateTriggers adds or removes Lambda triggers for a user pool. When a trigger
 is specified with a `nil` value,
// it is removed from the user pool.
func (actor CognitoActions) UpdateTriggers(ctx context.Context, userPoolId
 string, triggers ...TriggerInfo) error {
 output, err := actor.CognitoClient.DescribeUserPool(ctx,
 &cognitoidentityprovider.DescribeUserPoolInput{
  UserPoolId: aws.String(userPoolId),
 })
 if err != nil {
  log.Printf("Couldn't get info about user pool %v. Here's why: %v\n",
 userPoolId, err)
  return err
 }
 lambdaConfig := output.UserPool.LambdaConfig
 for _, trigger := range triggers {
  switch trigger.Trigger {
  case PreSignUp:
   lambdaConfig.PreSignUp = trigger.HandlerArn
  case UserMigration:
   lambdaConfig.UserMigration = trigger.HandlerArn
  case PostAuthentication:
   lambdaConfig.PostAuthentication = trigger.HandlerArn
  }
 }
 _, err = actor.CognitoClient.UpdateUserPool(ctx,
 &cognitoidentityprovider.UpdateUserPoolInput{
  UserPoolId:   aws.String(userPoolId),
  LambdaConfig: lambdaConfig,
 })
 if err != nil {
  log.Printf("Couldn't update user pool %v. Here's why: %v\n", userPoolId, err)
```

```
 }
 return err
}



// SignUp signs up a user with Amazon Cognito.
func (actor CognitoActions) SignUp(ctx context.Context, clientId string, userName
 string, password string, userEmail string) (bool, error) {
 confirmed := false
 output, err := actor.CognitoClient.SignUp(ctx,
 &cognitoidentityprovider.SignUpInput{
  ClientId: aws.String(clientId),
  Password: aws.String(password),
  Username: aws.String(userName),
  UserAttributes: []types.AttributeType{
   {Name: aws.String("email"), Value: aws.String(userEmail)},
  },
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't sign up user %v. Here's why: %v\n", userName, err)
  }
 } else {
  confirmed = output.UserConfirmed
 }
 return confirmed, err
}



// SignIn signs in a user to Amazon Cognito using a username and password
 authentication flow.
func (actor CognitoActions) SignIn(ctx context.Context, clientId string, userName
 string, password string) (*types.AuthenticationResultType, error) {
 var authResult *types.AuthenticationResultType
 output, err := actor.CognitoClient.InitiateAuth(ctx,
 &cognitoidentityprovider.InitiateAuthInput{
  AuthFlow:       "USER_PASSWORD_AUTH",
  ClientId:       aws.String(clientId),
  AuthParameters: map[string]string{"USERNAME": userName, "PASSWORD": password},
```

```
 })
 if err != nil {
  var resetRequired *types.PasswordResetRequiredException
  if errors.As(err, &resetRequired) {
   log.Println(*resetRequired.Message)
  } else {
   log.Printf("Couldn't sign in user %v. Here's why: %v\n", userName, err)
  }
 } else {
  authResult = output.AuthenticationResult
 }
 return authResult, err
}



// ForgotPassword starts a password recovery flow for a user. This flow typically
 sends a confirmation code
// to the user's configured notification destination, such as email.
func (actor CognitoActions) ForgotPassword(ctx context.Context, clientId string,
 userName string) (*types.CodeDeliveryDetailsType, error) {
 output, err := actor.CognitoClient.ForgotPassword(ctx,
 &cognitoidentityprovider.ForgotPasswordInput{
  ClientId: aws.String(clientId),
  Username: aws.String(userName),
 })
 if err != nil {
  log.Printf("Couldn't start password reset for user '%v'. Here;s why: %v\n",
 userName, err)
 }
 return output.CodeDeliveryDetails, err
}



// ConfirmForgotPassword confirms a user with a confirmation code and a new
 password.
func (actor CognitoActions) ConfirmForgotPassword(ctx context.Context, clientId
 string, code string, userName string, password string) error {
 _, err := actor.CognitoClient.ConfirmForgotPassword(ctx,
 &cognitoidentityprovider.ConfirmForgotPasswordInput{
  ClientId:         aws.String(clientId),
  ConfirmationCode: aws.String(code),
  Password:         aws.String(password),
```

```
  Username:         aws.String(userName),
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't confirm user %v. Here's why: %v", userName, err)
  }
 }
 return err
}



// DeleteUser removes a user from the user pool.
func (actor CognitoActions) DeleteUser(ctx context.Context, userAccessToken
 string) error {
 _, err := actor.CognitoClient.DeleteUser(ctx,
 &cognitoidentityprovider.DeleteUserInput{
  AccessToken: aws.String(userAccessToken),
 })
 if err != nil {
  log.Printf("Couldn't delete user. Here's why: %v\n", err)
 }
 return err
}



// AdminCreateUser uses administrator credentials to add a user to a user pool.
 This method leaves the user
// in a state that requires they enter a new password next time they sign in.
func (actor CognitoActions) AdminCreateUser(ctx context.Context, userPoolId
 string, userName string, userEmail string) error {
 _, err := actor.CognitoClient.AdminCreateUser(ctx,
 &cognitoidentityprovider.AdminCreateUserInput{
  UserPoolId:     aws.String(userPoolId),
  Username:       aws.String(userName),
  MessageAction:  types.MessageActionTypeSuppress,
  UserAttributes: []types.AttributeType{{Name: aws.String("email"), Value:
 aws.String(userEmail)}},
 })
 if err != nil {
```

```
  var userExists *types.UsernameExistsException
  if errors.As(err, &userExists) {
   log.Printf("User %v already exists in the user pool.", userName)
   err = nil
  } else {
   log.Printf("Couldn't create user %v. Here's why: %v\n", userName, err)
  }
 }
 return err
}



// AdminSetUserPassword uses administrator credentials to set a password for a
 user without requiring a
// temporary password.
func (actor CognitoActions) AdminSetUserPassword(ctx context.Context, userPoolId
 string, userName string, password string) error {
 _, err := actor.CognitoClient.AdminSetUserPassword(ctx,
 &cognitoidentityprovider.AdminSetUserPasswordInput{
  Password:   aws.String(password),
  UserPoolId: aws.String(userPoolId),
  Username:   aws.String(userName),
  Permanent:  true,
 })
 if err != nil {
  var invalidPassword *types.InvalidPasswordException
  if errors.As(err, &invalidPassword) {
   log.Println(*invalidPassword.Message)
  } else {
   log.Printf("Couldn't set password for user %v. Here's why: %v\n", userName,
 err)
  }
 }
 return err
}
```

创建一个封装 DynamoDB 操作的结构。

```
import (
```

```
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/feature/dynamodb/attributevalue"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb"
 "github.com/aws/aws-sdk-go-v2/service/dynamodb/types"
)

// DynamoActions encapsulates the Amazon Simple Notification Service (Amazon SNS)
 actions
// used in the examples.
type DynamoActions struct {
 DynamoClient *dynamodb.Client
}

// User defines structured user data.
type User struct {
 UserName  string
 UserEmail string
 LastLogin *LoginInfo `dynamodbav:",omitempty"`
}

// LoginInfo defines structured custom login data.
type LoginInfo struct {
 UserPoolId string
 ClientId   string
 Time       string
}

// UserList defines a list of users.
type UserList struct {
 Users []User
}

// UserNameList returns the usernames contained in a UserList as a list of
 strings.
func (users *UserList) UserNameList() []string {
 names := make([]string, len(users.Users))
 for i := 0; i < len(users.Users); i++ {
  names[i] = users.Users[i].UserName
 }
 return names
```

```
}

// PopulateTable adds a set of test users to the table.
func (actor DynamoActions) PopulateTable(ctx context.Context, tableName string)
 error {
 var err error
 var item map[string]types.AttributeValue
 var writeReqs []types.WriteRequest
 for i := 1; i < 4; i++ {
  item, err = attributevalue.MarshalMap(User{UserName: fmt.Sprintf("test_user_
%v", i), UserEmail: fmt.Sprintf("test_email_%v@example.com", i)})
  if err != nil {
   log.Printf("Couldn't marshall user into DynamoDB format. Here's why: %v\n",
 err)
   return err
  }
  writeReqs = append(writeReqs, types.WriteRequest{PutRequest:
 &types.PutRequest{Item: item}})
 }
 _, err = actor.DynamoClient.BatchWriteItem(ctx, &dynamodb.BatchWriteItemInput{
  RequestItems: map[string][]types.WriteRequest{tableName: writeReqs},
 })
 if err != nil {
  log.Printf("Couldn't populate table %v with users. Here's why: %v\n",
 tableName, err)
 }
 return err
}

// Scan scans the table for all items.
func (actor DynamoActions) Scan(ctx context.Context, tableName string) (UserList,
 error) {
 var userList UserList
 output, err := actor.DynamoClient.Scan(ctx, &dynamodb.ScanInput{
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't scan table %v for items. Here's why: %v\n", tableName,
 err)
 } else {
  err = attributevalue.UnmarshalListOfMaps(output.Items, &userList.Users)
  if err != nil {
   log.Printf("Couldn't unmarshal items into users. Here's why: %v\n", err)
  }
```

```
 }
 return userList, err
}

// AddUser adds a user item to a table.
func (actor DynamoActions) AddUser(ctx context.Context, tableName string, user
 User) error {
 userItem, err := attributevalue.MarshalMap(user)
 if err != nil {
  log.Printf("Couldn't marshall user to item. Here's why: %v\n", err)
 }
 _, err = actor.DynamoClient.PutItem(ctx, &dynamodb.PutItemInput{
  Item:      userItem,
  TableName: aws.String(tableName),
 })
 if err != nil {
  log.Printf("Couldn't put item in table %v. Here's why: %v", tableName, err)
 }
 return err
}
```

创建一个封装 CloudWatch 日志操作的结构。

```
import (
 "context"
 "fmt"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs"
 "github.com/aws/aws-sdk-go-v2/service/cloudwatchlogs/types"
)

type CloudWatchLogsActions struct {
 CwlClient *cloudwatchlogs.Client
}

// GetLatestLogStream gets the most recent log stream for a Lambda function.
func (actor CloudWatchLogsActions) GetLatestLogStream(ctx context.Context,
 functionName string) (types.LogStream, error) {
```

```
 var logStream types.LogStream
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.DescribeLogStreams(ctx,
 &cloudwatchlogs.DescribeLogStreamsInput{
  Descending:   aws.Bool(true),
  Limit:        aws.Int32(1),
  LogGroupName: aws.String(logGroupName),
  OrderBy:      types.OrderByLastEventTime,
 })
 if err != nil {
  log.Printf("Couldn't get log streams for log group %v. Here's why: %v\n",
 logGroupName, err)
 } else {
  logStream = output.LogStreams[0]
 }
 return logStream, err
}

// GetLogEvents gets the most recent eventCount events from the specified log
 stream.
func (actor CloudWatchLogsActions) GetLogEvents(ctx context.Context, functionName
 string, logStreamName string, eventCount int32) (
 []types.OutputLogEvent, error) {
 var events []types.OutputLogEvent
 logGroupName := fmt.Sprintf("/aws/lambda/%s", functionName)
 output, err := actor.CwlClient.GetLogEvents(ctx,
 &cloudwatchlogs.GetLogEventsInput{
  LogStreamName: aws.String(logStreamName),
  Limit:         aws.Int32(eventCount),
  LogGroupName:  aws.String(logGroupName),
 })
 if err != nil {
  log.Printf("Couldn't get log event for log stream %v. Here's why: %v\n",
 logStreamName, err)
 } else {
  events = output.Events
 }
 return events, err
}
```

创建一个封装动作的结构。 Amazon CloudFormation

```
import (
 "context"
 "log"

 "github.com/aws/aws-sdk-go-v2/aws"
 "github.com/aws/aws-sdk-go-v2/service/cloudformation"
)

// StackOutputs defines a map of outputs from a specific stack.
type StackOutputs map[string]string

type CloudFormationActions struct {
 CfnClient *cloudformation.Client
}

// GetOutputs gets the outputs from a CloudFormation stack and puts them into a
 structured format.
func (actor CloudFormationActions) GetOutputs(ctx context.Context, stackName
 string) StackOutputs {
 output, err := actor.CfnClient.DescribeStacks(ctx,
 &cloudformation.DescribeStacksInput{
  StackName: aws.String(stackName),
 })
 if err != nil || len(output.Stacks) == 0 {
  log.Panicf("Couldn't find a CloudFormation stack named %v. Here's why: %v\n",
 stackName, err)
 }
 stackOutputs := StackOutputs{}
 for _, out := range output.Stacks[0].Outputs {
  stackOutputs[*out.OutputKey] = *out.OutputValue
 }
 return stackOutputs
}
```

清理资源。

```
import (
 "context"
 "log"
```

```
 "user_pools_and_lambda_triggers/actions"

 "github.com/awsdocs/aws-doc-sdk-examples/gov2/demotools"
)

// Resources keeps track of AWS resources created during an example and handles
// cleanup when the example finishes.
type Resources struct {
 userPoolId       string
 userAccessTokens []string
 triggers         []actions.Trigger

 cognitoActor *actions.CognitoActions
 questioner   demotools.IQuestioner
}

func (resources *Resources) init(cognitoActor *actions.CognitoActions, questioner
 demotools.IQuestioner) {
 resources.userAccessTokens = []string{}
 resources.triggers = []actions.Trigger{}
 resources.cognitoActor = cognitoActor
 resources.questioner = questioner
}

// Cleanup deletes all AWS resources created during an example.
func (resources *Resources) Cleanup(ctx context.Context) {
 defer func() {
  if r := recover(); r != nil {
   log.Printf("Something went wrong during cleanup.\n%v\n", r)
   log.Println("Use the AWS Management Console to remove any remaining resources
 \n" +
    "that were created for this scenario.")
  }
 }()

 wantDelete := resources.questioner.AskBool("Do you want to remove all of the AWS
 resources that were created "+
  "during this demo (y/n)?", "y")
 if wantDelete {
  for _, accessToken := range resources.userAccessTokens {
   err := resources.cognitoActor.DeleteUser(ctx, accessToken)
   if err != nil {
    log.Println("Couldn't delete user during cleanup.")
    panic(err)
```

```
    }
    log.Println("Deleted user.")
   }
   triggerList := make([]actions.TriggerInfo, len(resources.triggers))
   for i := 0; i < len(resources.triggers); i++ {
    triggerList[i] = actions.TriggerInfo{Trigger: resources.triggers[i],
  HandlerArn: nil}
   }
   err := resources.cognitoActor.UpdateTriggers(ctx, resources.userPoolId,
  triggerList...)
   if err != nil {
    log.Println("Couldn't update Cognito triggers during cleanup.")
    panic(err)
   }
   log.Println("Removed Cognito triggers from user pool.")
  } else {
   log.Println("Be sure to remove resources when you're done with them to avoid
  unexpected charges!")
  }
 }
```

- 有关 API 详细信息，请参阅《适用于 Go 的 Amazon SDK API 参考》中的以下主题。
  - AdminCreateUser
  - AdminSetUserPassword
  - DeleteUser
  - InitiateAuth
  - UpdateUserPool

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

使用 Amazon Cognito 同步的代码示例 Amazon SDKs

以下代码示例展示了如何将 Amazon Cognito Sync 与 Amazon 软件开发套件 (SDK) 一起使用。

操作是大型程序的代码摘录，必须在上下文中运行。您可以通过操作了解如何调用单个服务函数，还可以通过函数相关场景的上下文查看操作。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

代码示例

- 使用 Amazon Cognito 同步的基本示例 Amazon SDKs
  - 使用 Amazon Cognito 同步的操作 Amazon SDKs
    - 与 Amazon SDK ListIdentityPoolUsage 配合使用

使用 Amazon Cognito 同步的基本示例 Amazon SDKs

以下代码示例展示了如何使用 Amazon Cognito Sync 的基础知识。 Amazon SDKs

示例

- 使用 Amazon Cognito 同步的操作 Amazon SDKs
  - 与 Amazon SDK ListIdentityPoolUsage 配合使用

使用 Amazon Cognito 同步的操作 Amazon SDKs

以下代码示例演示了如何使用执行各个 Amazon Cognito 同步操作。 Amazon SDKs每个示例都包含一个指向的链接 GitHub，您可以在其中找到有关设置和运行代码的说明。

以下示例仅包括最常用的操作。有关完整列表，请参阅 Amazon Cognito 同步 API 参考。

示例

- 与 Amazon SDK ListIdentityPoolUsage 配合使用

与 Amazon SDK **`ListIdentityPoolUsage`** 配合使用

以下代码示例演示如何使用 `ListIdentityPoolUsage`。

Rust

适用于 Rust 的 SDK

> **Note**
>
> 还有更多相关信息 GitHub。查找完整示例，学习如何在 Amazon 代码示例存储库中进行设置和运行。

```
async fn show_pools(client: &Client) -> Result<(), Error> {
    let response = client
        .list_identity_pool_usage()
        .max_results(10)
        .send()
        .await?;

    let pools = response.identity_pool_usages();
    println!("Identity pools:");

    for pool in pools {
        println!(
            "  Identity pool ID:    {}",
            pool.identity_pool_id().unwrap_or_default()
        );
        println!(
            "  Data storage:        {}",
            pool.data_storage().unwrap_or_default()
        );
        println!(
            "  Sync sessions count: {}",
            pool.sync_sessions_count().unwrap_or_default()
        );
        println!(
            "  Last modified:       {}",
            pool.last_modified_date().unwrap().to_chrono_utc()?
        );
        println!();
    }

    println!("Next token: {}", response.next_token().unwrap_or_default());
```

```
    Ok(())
}
```

- 有关 API 的详细信息，请参阅适用ListIdentityPoolUsage于 Rust 的Amazon SDK API 参考。

有关 S Amazon DK 开发者指南和代码示例的完整列表，请参阅将此服务与 Amazon SDK 配合使用。本主题还包括有关入门的信息以及有关先前的 SDK 版本的详细信息。

多租户应用程序最佳实践

Amazon Cognito 用户池适用于生成大量请求的多租户应用，这些请求必须保持在 Amazon Cognito 的配额范围内。要在客户群增长时纵向扩展此容量，您可以购买额外的配额容量。

> **Note**
>
> Amazon Cognito 配额按和适用。 Amazon Web Services 账户 Amazon Web Services 区域这些配额在应用程序中的所有租户之间共享。请查看 Amazon Cognito 服务配额，并确保配额符合您应用程序中的预期容量和预期租户数量。

本节介绍您可以实施的方法来区分同一区域内的 Amazon Cognito 资源和。 Amazon Web Services 账户您也可以将租户分配到多个 Amazon Web Services 账户 或多个区域，并为每个租户分配自己的配额。多区域多租赁的其他优势包括尽可能高的隔离级别、为全球分布的用户实现更短的网络传输时间，以及遵守组织中现有的分发模式。

单区域多租赁也可为客户和管理员带来好处。

以下列表介绍了使用共享资源的多租赁的一些优势。

多租赁的优势

通用用户目录

多租赁支持客户在多个应用程序中拥有账户的模式。您可以将来自第三方提供者的身份关联到单个一致的用户池配置文件中。如果用户配置文件是其租户所独有的，则任何具有单个用户池的多租赁策略都有一个用户管理的切入点。

通用安全性

在共享用户池中，您可以创建单一的安全标准，并将相同的威胁防护、多因素身份验证 (MFA) Amazon WAF和标准应用于所有租户。由于 Amazon WAF Web ACL 必须与您关联的资源 Amazon Web Services 区域 相同，因此多租户提供对复杂资源的共享访问权限。如果您想在多区域 Amazon Cognito 应用程序中保持一致的安全配置，则必须应用操作标准，在资源之间复制配置。

通用自定义

您可以使用自定义用户池和身份池 Amazon Lambda。用户池中的 Lambda 触发器和身份池中的 Amazon Cognito 事件的配置会变得复杂。Lambda 函数必须与您的用户池或身份池位于同一个

Amazon Web Services 区域 中。共享 Lambda 函数可为一个区域内的自定义身份验证流程、用户迁移、令牌生成和其他功能强制执行标准。

通用消息发送

Amazon Simple Notification Service（Amazon SNS）需要在区域中进行额外配置，然后才能向用户发送短信。您可以使用 Amazon Simple Email Service（Amazon SES）已验证的身份和区域中包含的域来发送电子邮件消息。

借助多租赁，您可以在所有租户之间共享此配置和维护开销。由于 Amazon SNS 和 Amazon SES 并非在所有 Amazon Web Services 区域均可用，因此在不同区域之间分配资源需要额外考量。

使用自定义消息传递提供方时，您可以通过单个 Lambda 函数的通用自定义来管理您的消息传送。

托管登录在浏览器中设置会话 Cookie，以便识别已经通过身份验证的用户。当您对用户池中的本地用户进行身份验证时，用户的会话 Cookie 会在同一用户池中针对所有应用程序客户端对用户进行身份验证。本地用户仅存在于您的用户群体目录中，无需通过外部 IdP 进行联合身份验证。会话 Cookie 的有效期为 1 小时。您无法更改会话 Cookie 的持续时间。

有两种方法可以防止使用托管 UI 会话 Cookie 跨应用程序客户端登录。

- 将用户分配到按租户划分的用户池。
- 将托管 UI 登录替换为 Amazon Cognito 用户池 API 登录。

主题

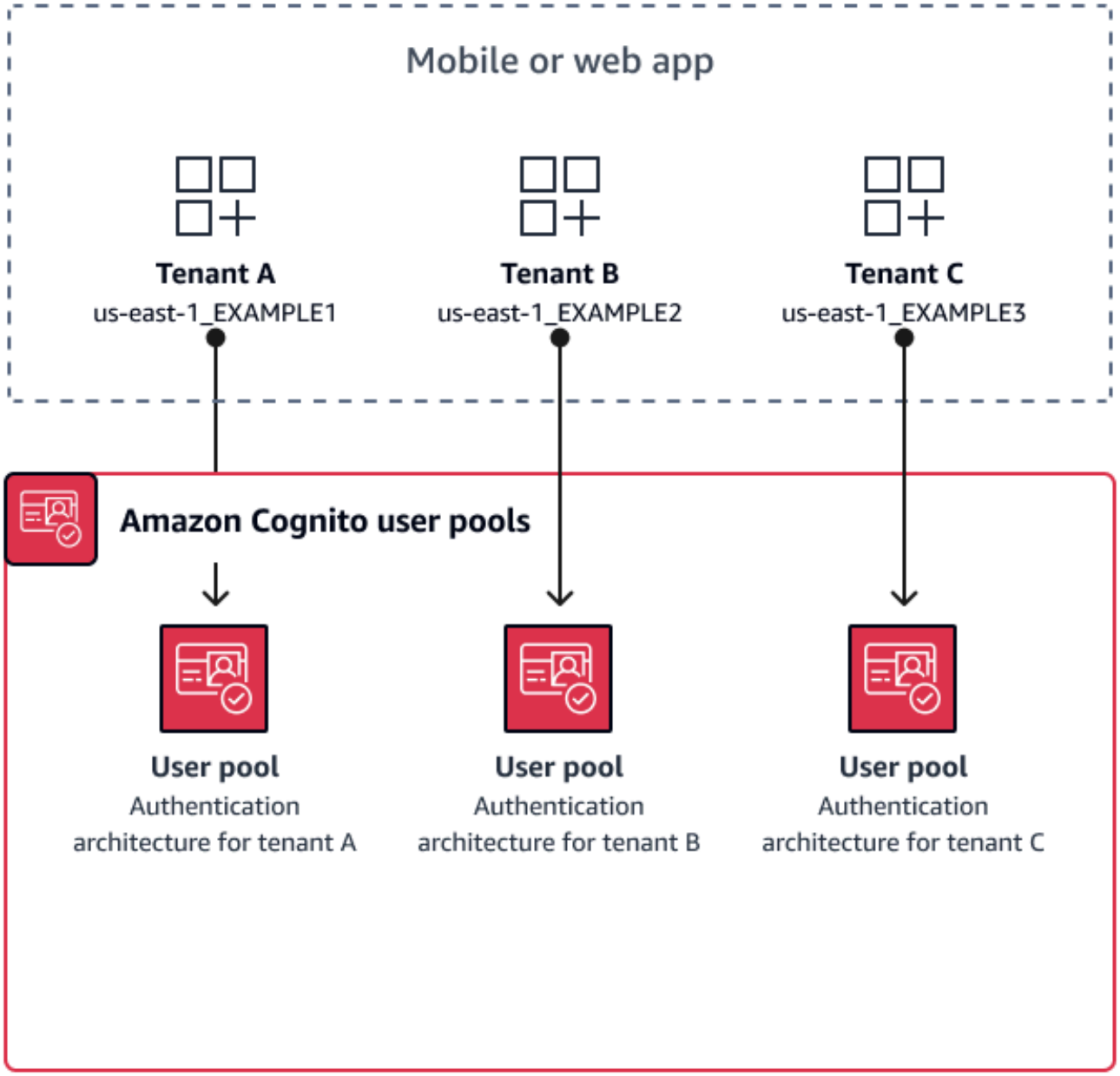
- 用户池多租赁最佳实践
- 应用程序客户端多租赁最佳实践
- 用户组多租赁最佳实践
- 自定义属性多租赁最佳实践
- 自定义范围多租赁最佳实践
- 多租户安全建议

用户池多租赁最佳实践

在应用程序中为每个租户创建一个用户池。此方法为每个租户提供最大程度的隔离。您可以为每个租户实施不同的配置。通过用户池进行租户隔离使您可以灵活地进行 user-to-tenant映射。您可以为同一用户创建多个配置文件。但是，每个用户必须为他们可以访问的每个租户单独注册。

使用此方法，可以单独为每个租户设置托管 UI，并将用户重新导向到您应用程序的租户特定实例。您还可以使用此方法与 Amazon API Gateway 等后端服务集成。

下图显示每个租户都有一个专用用户池。



何时实施用户池多租赁

当隔离和自定义是您的主要关注点时。在具有多个用户池的架构中，用户和租户之间的关系可能很复杂。举一个例子，您有两个教育租户。同一个用户在一个应用程序中可能是访问受限的学生，而在另一个应用程序中可能是具有较高权限的教师。您可能需要在一个应用程序中使用 MFA，但在另一个应用程序中不需要使用 MFA，或者两个应用程序有不同的密码策略。由于本地用户可以使用托管登录登录登录用户池中的多个应用程序客户端，因此当您希望多个租户使用托管登录登录时，用户池多租户也是理想的选择。

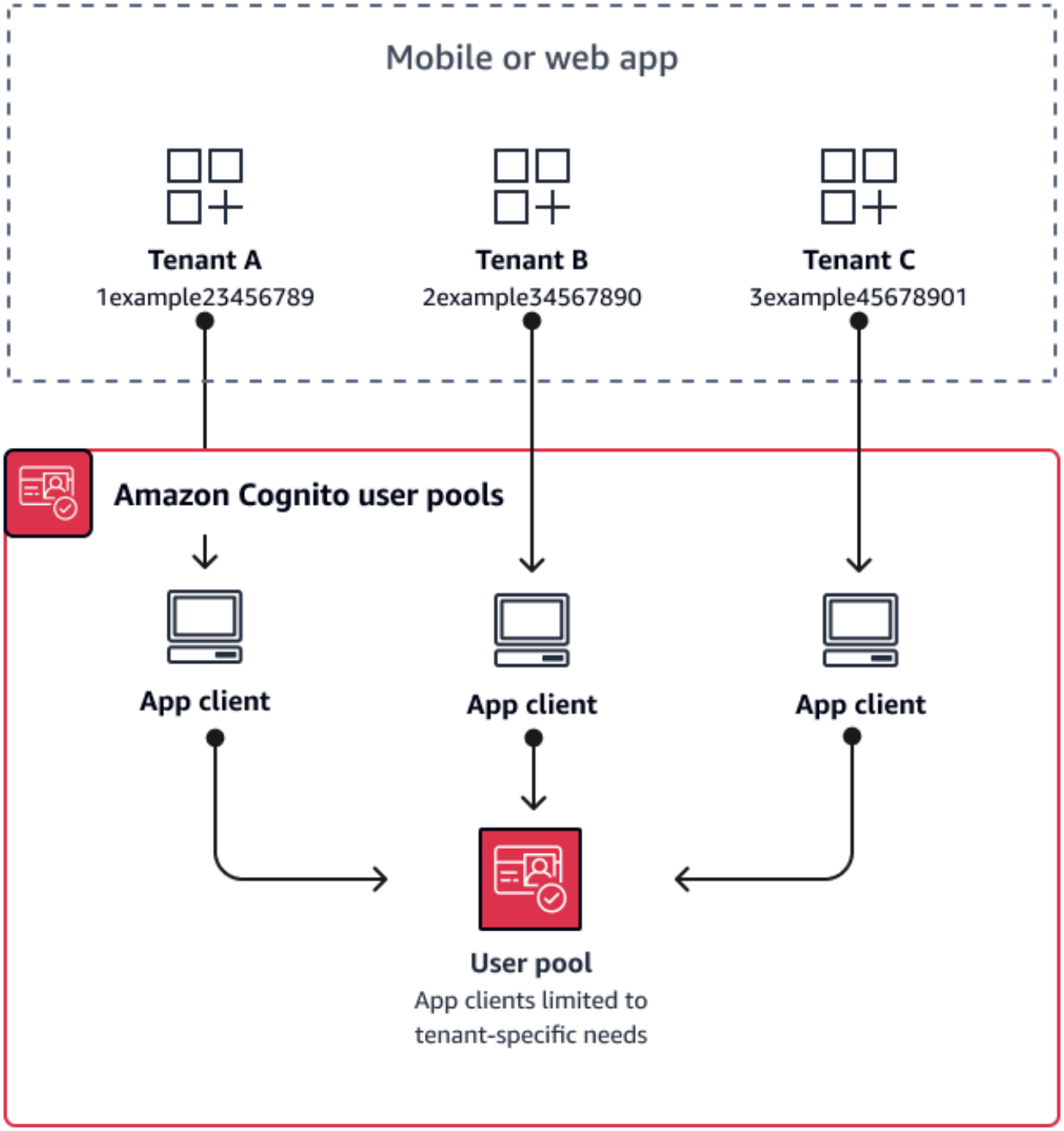
工作量水平

使用此方法的开发和操作工作量很大。为了确保您的应用程序系列获得一致且可预测的结果，必须将 Amazon Cognito 资源与自动化工具集成，并在身份验证架构变得越来越复杂时维持最初设定的基准。当您想为应用程序打造统一的起始页面时，您必须构建用户界面（UI）元素来捕获用户的初步选择，并将他们引导到正确的资源。

应用程序客户端多租赁最佳实践

为应用程序中的每个租户创建一个应用程序客户端。借助应用程序客户端多租赁，您可以将任何用户分配给与租户关联的应用程序客户端，并保留单个用户配置文件。由于您可以将用户池中的任何或所有身份提供者 (IdPs) 分配给应用程序客户端，因此租户应用程序客户端可以允许使用租户特定的 IdP 登录。当用户存在于多个租户中时，您可以将他们的个人资料与多个租户关联起来， IdPs 以获得一致的用户体验。

下图显示共享用户池中的每个租户有一个专用应用程序客户端。



何时实施应用程序客户端多租赁

当您可在用户池级别选择设置的通用配置（例如 Lambda 触发器、密码策略以及电子邮件和短信的内容和传递方式）时。由于共享用户池中的用户可以登录任何应用程序客户端，因此应用程序客户端多租

户非常适合使用或 A app-client-specific IdPs mazon Cognito 用户池 API 登录。App-Client 多租户也非常适合您希望允许用户在多个应用程序之间切换的 one-to-many环境。

工作量水平

实施应用程序客户端多租赁需要一定的工作量。应用程序客户端多租赁面临的一个主要挑战是，租户能否提供托管 UI Cookie 并在应用程序之间切换。在应用程序客户端多租赁架构中，在需要隔离的地方避免使用托管 UI 登录。您可以使用内置的应用程序客户端逻辑分发移动应用程序或指向 Web 应用程序的链接，也可以构建初步的 UI 元素，用来确定用户的租赁。因为您无需跨多个用户池和身份池来标准化和维护配置，所以工作量较低。

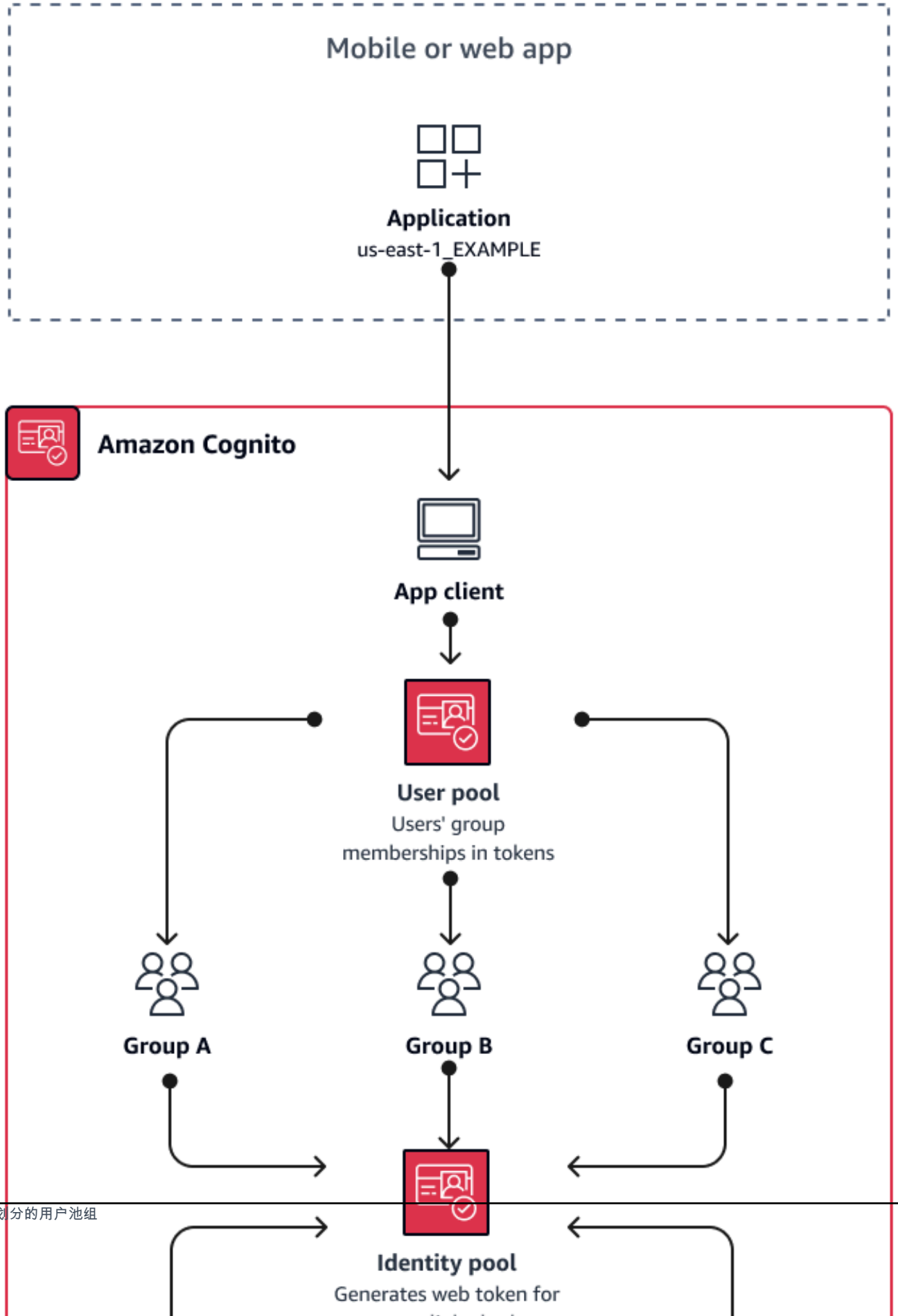
用户组多租赁最佳实践

当您的架构需要 Amazon Cognito 用户池和身份池时，基于组的多租赁效果更好。

用户池 ID 令牌和访问令牌包含 `cognito:groups` 声明。此外，ID 令牌包含 `cognito:roles` 和 `cognito:preferred_role` 声明。当您的应用程序中身份验证的主要结果是来自身份池的临时 Amazon 凭证时，您的用户的组成员资格可以决定他们获得的 IAM 角色和权限。

例如，假设三个租户，每个租户都在自己的 Amazon S3 存储桶中存储应用程序资产。将每个租户的用户分配到关联的组，为该组配置首选角色，并授予该角色读取其存储桶的权限。

下图显示租户共享应用程序客户端和用户池，且用户池中的专用组用来确定他们是否有资格代入 IAM 角色。


Mobile or web app
Application
us-east-1_EXAMPLE
Amazon Cognito
App client
User pool
Users' group
memberships in tokens
Group A
Group B
Group C
Identity pool
Generates web token for

何时实施组多租赁

当 Amazon 资源访问是您的首要考虑时。Amazon Cognito 用户池中的组是一种基于角色的访问控制（RBAC）机制。您可以在用户池中配置许多组，并根据组优先级作出复杂的 RBAC 决策。身份池可以为优先级最高的角色、组声明中的任何角色或来自用户令牌中其他声明的角色分配凭证。

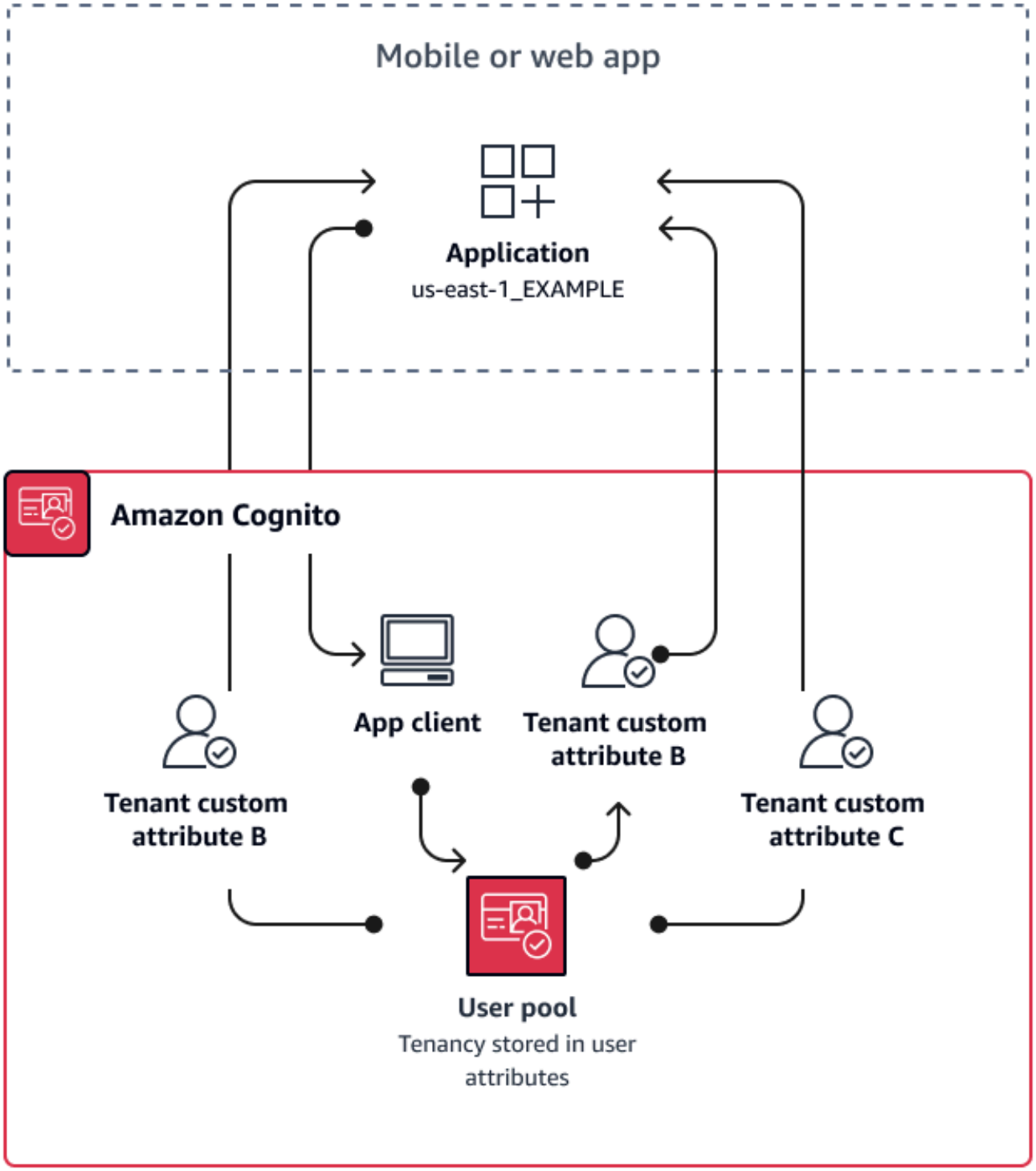
工作量水平

仅通过组成员身份来维持多租赁架构的工作量很低。但是，要将用户池组的角色扩展到内置的 IAM 角色选择容量之外，必须构建应用程序逻辑来处理用户令牌中的组成员身份，并确定要在客户端中执行的操作。可以将 Amazon Verified Permissions 与应用程序集成，从而作出客户端授权决策。当前未在 Verified Permissions IsAuthorizedWithTokenAPI 操作中处理群组标识符，但您可以开发解析令牌内容的自定义代码，包括群组成员资格声明。

自定义属性多租赁最佳实践

Amazon Cognito 支持使用自选名称的自定义属性。自定义属性在一种情况下很有用，那就是当它们用于区分共享用户池中用户的租赁时。当您为用户分配一个像 `custom:tenantID` 这样的属性值时，您的应用程序可以相应地对租户特定的资源分配访问权限。定义租户 ID 的自定义属性对应用程序客户端来说应该是不可变或只读的。

下图显示租户共享应用程序客户端和用户池，用户池中的自定义属性指明他们所属的租户。



当自定义属性确定租赁时，您可以分发单个应用程序或登录 URL。用户登录后，应用程序可以处理 `custom:tenantID` 声明，确定要加载哪些资产、要应用的品牌以及要显示的特征。要根据用户属性

作出高级访问控制决策，请在 Amazon Verified Permissions 中将用户池设置为身份提供者，然后根据 ID 令牌或访问令牌的内容生成访问决策。

何时实施自定义属性多租赁

当租赁是表面层级时。租户属性有助于实现品牌和布局结果。当您想在租户之间实现显著隔离时，自定义属性并不是最佳选择。如果租户之间有任何差异必须在用户池或应用程序客户端级别进行配置（例如 MFA 或托管 UI 品牌），则您需要以自定义属性无法实现的方式在租户级别进行区分。借助身份池，您甚至可以根据用户在其 ID 令牌中的自定义属性声明，为用户选择相应的 IAM 角色。

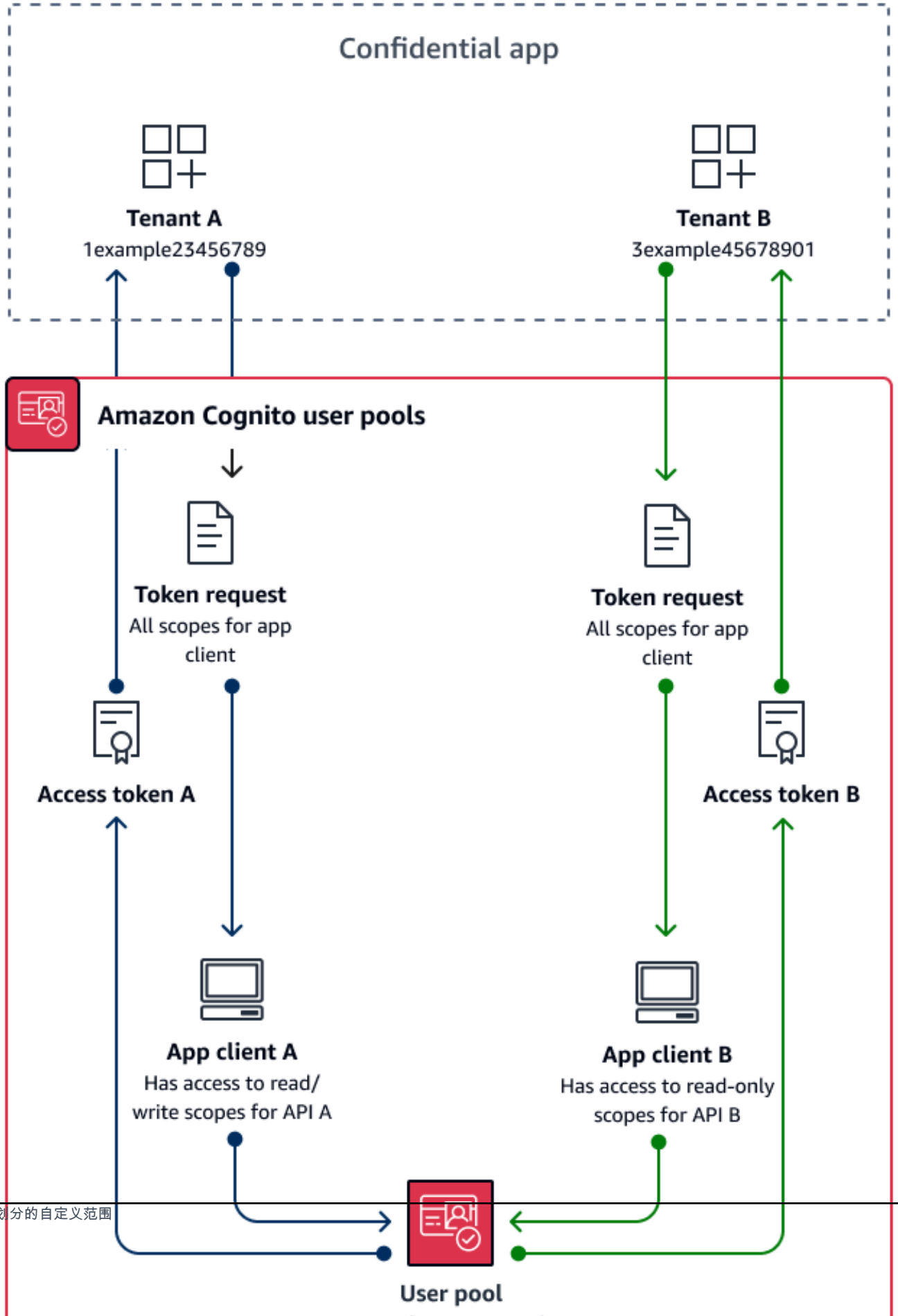
工作量水平

由于自定义属性多租赁将基于租户的授权决策职责转移到应用程序上，因此工作量往往很高。如果您已经精通用于解析 OIDC 声明的客户端配置或熟练使用 Amazon Verified Permissions，那么这种方法所需的工作量可能更少。

自定义范围多租赁最佳实践

Amazon Cognito 支持资源服务器的自定义 OAuth 2.0 范围。您可以在用户池中实现具有自定义范围的 machine-to-machine (M2M) 授权模型的应用程序客户端多租户。基于范围的多租赁可以在应用程序客户端或应用程序配置中定义访问权限，从而可减少实施 M2M 多租赁所需的工作量。

下图说明了自定义范围多租赁的一个选项。该图显示，每个租户都有一个专用的应用程序客户端，可以访问用户池中的相关范围。


Confidential app
Tenant A
1example23456789
Tenant B
3example45678901
Amazon Cognito user pools
Token request
All scopes for app client
Token request
All scopes for app client
Access token A
Access token B
App client A
Has access to read/ write scopes for API A
App client B
Has access to read-only scopes for API B
User pool

何时实施自定义范围多租赁

当您的使用场景是机器到机器（M2M）授权，且机密客户端使用客户端凭证来进行授权时。作为一项最佳实践，请创建应用程序客户端专用的资源服务器。自定义范围多租赁可以取决于请求，也可以取决于客户端。

取决于请求

实施应用程序逻辑，仅请求符合租户要求的范围。例如，应用程序客户端也许可以授予对 API A 和 API B 的读取和写入权限，但租户应用程序 A 仅请求 API A 的读取范围和表示租赁的范围。利用此模型可对租户之间的共享范围进行更复杂的组合。

取决于客户端

在授权请求中请求分配给应用程序客户端的所有范围。为此，可以在向 令牌端点 发出的请求中忽略 `scope` 请求参数。利用此模型可让应用程序客户端存储您要添加到自定义范围的访问指示器。

无论是哪种情况，应用程序收到的访问令牌中的范围都会表明它们对依赖的数据来源所具备的权限。范围还可以向您的应用程序提供其他信息：

- 指定租赁
- 参与请求日志的记录
- 指明 APIs 该应用程序已被授权查询
- 告知关于活跃客户的初步检查。

工作量水平

根据应用程序的规模，自定义范围多租赁需要不同的工作量。必须将应用程序逻辑设计为允许应用程序解析访问令牌并发出相应的 API 请求。

例如，资源服务器范围的格式为 `[resource server identifier]/[name]`。资源服务器标识符不太可能与租户范围的授权决策相关，因此需要一致地解析范围名称。

示例资源

以下 Amazon CloudFormation 模板使用一个资源服务器和一个应用程序客户端为自定义范围的多租户创建用户池。

```
AWSTemplateFormatVersion: "2010-09-09"
Description: A sample template illustrating scope-based multi-tenancy
```

```
Resources:
  MyUserPool:
    Type: "AWS::Cognito::UserPool"
  MyUserPoolDomain:
    Type: AWS::Cognito::UserPoolDomain
    Properties:
      UserPoolId: !Ref MyUserPool
      # Note that the value for "Domain" must be unique across all of AWS.
      # In production, you may want to consider using a custom domain.
      # See: https://docs.aws.amazon.com/cognito/latest/developerguide/cognito-user-
pools-add-custom-domain.html#cognito-user-pools-add-custom-domain-adding
      Domain: !Sub "example-userpool-domain-${AWS::AccountId}"
  MyUserPoolResourceServer:
    Type: "AWS::Cognito::UserPoolResourceServer"
    Properties:
      Identifier: resource1
      Name: resource1
      Scopes:
        - ScopeDescription: Read-only access
          ScopeName: readScope
      UserPoolId: !Ref MyUserPool
  MyUserPoolTenantBatch1ResourceServer:
    Type: "AWS::Cognito::UserPoolResourceServer"
    Properties:
      Identifier: TenantBatch1
      Name: TenantBatch1
      Scopes:
        - ScopeDescription: tenant1 identifier
          ScopeName: tenant1
        - ScopeDescription: tenant2 identifier
          ScopeName: tenant2
      UserPoolId: !Ref MyUserPool
  MyUserPoolClientTenant1:
    Type: "AWS::Cognito::UserPoolClient"
    Properties:
      AllowedOAuthFlows:
        - client_credentials
      AllowedOAuthFlowsUserPoolClient: true
      AllowedOAuthScopes:
        - !Sub "${MyUserPoolTenantBatch1ResourceServer}/tenant1"
        - !Sub "${MyUserPoolResourceServer}/readScope"
      GenerateSecret: true
      UserPoolId: !Ref MyUserPool
Outputs:
```

```
  UserPoolClientId:
    Description: User pool client ID
    Value: !Ref MyUserPoolClientTenant1
  UserPoolDomain:
    Description: User pool domain
    Value: !Sub "https://${MyUserPoolDomain}.auth.${AWS::Region}.amazoncognito.com"
```

多租户安全建议

为了帮助提高应用程序的安全性，我们有下列建议：

- 使用 Amazon Verified Permissions 在您的应用程序中验证租赁。在允许用户在应用程序中提出请求之前，请先制定政策，检查用户池、应用程序客户端、群组或自定义属性权限。 Amazon 创建了经过验证的权限身份源，同时考虑了 Amazon Cognito 用户池。Verified Permissions 为多租赁管理提供了额外指导。
- 仅使用经过验证的电子邮件地址，根据域匹配授权用户访问租户。仅信任经过您的应用程序验证或者或者外部 IdP 提供了验证证明的电子邮件地址和电话号码。有关设置这些权限的更多详细信息，请参阅属性权限和范围。
- 对标识租户的用户配置文件属性使用不可变或只读自定义属性。只有在创建用户或用户在用户池中注册时，才能设置不可变属性的值。此外，向应用程序客户端授予对属性的只读访问权。
- 在租户的外部 IdP 与应用程序客户端之间使用 1:1 映射，以防止未经授权的跨租户访问。已通过外部 IdP 身份验证且具有有效 Amazon Cognito 会话 Cookie 的用户，可以访问信任相同 IdP 的其他租户应用程序。
- 在应用程序中实施与租户匹配的授权逻辑时，请限制用户，使他们不能修改用于授权用户访问租户的条件。此外，如果使用外部 IdP 进行联合身份验证，请限制租户身份提供商管理员，使其无法修改用户访问权限。

Amazon Cognito 常见场景

本主题介绍使用 Amazon Cognito 的六个常见场景。

Amazon Cognito 的两个主要组件是用户池和身份池。用户池是为您的 Web 和移动应用程序用户提供注册和登录选项的用户目录。身份池提供临时 Amazon 证书，以授予您的用户访问其他人的权限 Amazon Web Services 服务。

用户池是 Amazon Cognito 中的用户目录。您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供者（IdP）进行联合身份验证。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs无论您的用户是直接登录还是通过第三方登录，用户池的所有成员都有一个可通过开发工具包访问的目录配置文件。

借助身份池，您的用户可以获得访问 Amazon 服务（例如 Amazon S3 和 DynamoDB）的临时 Amazon 证书。身份池支持匿名访客用户，也支持通过第三方进行联合 IdPs。

主题

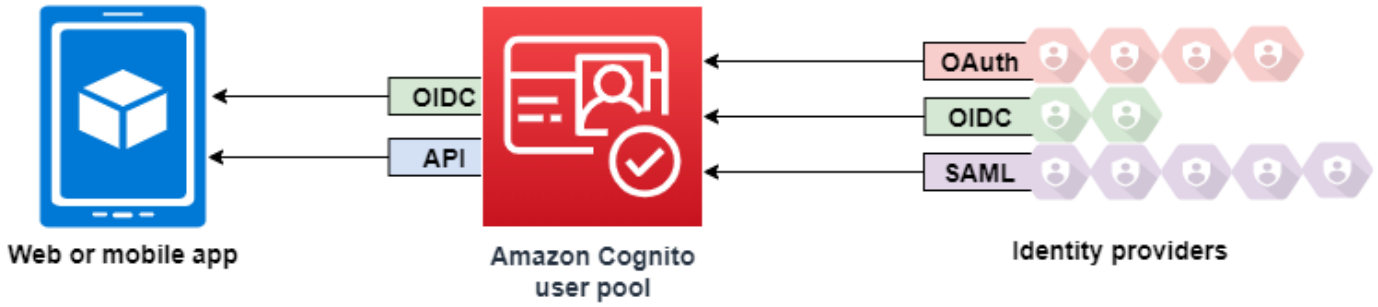
- 使用用户池进行身份验证
- 使用用户池令牌访问后端资源
- 将 API Gateway 和 Lambda 与用户池结合使用来访问资源
- 使用用户池和身份池访问 Amazon 服务
- 借助第三方进行身份验证并使用身份池访问 Amazon 服务
- 使用 Amazon Cognito 访问 Amazon AppSync 资源

使用用户池进行身份验证

您可以允许您的用户使用用户池进行身份验证。您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供者（IdP）进行联合身份验证。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs

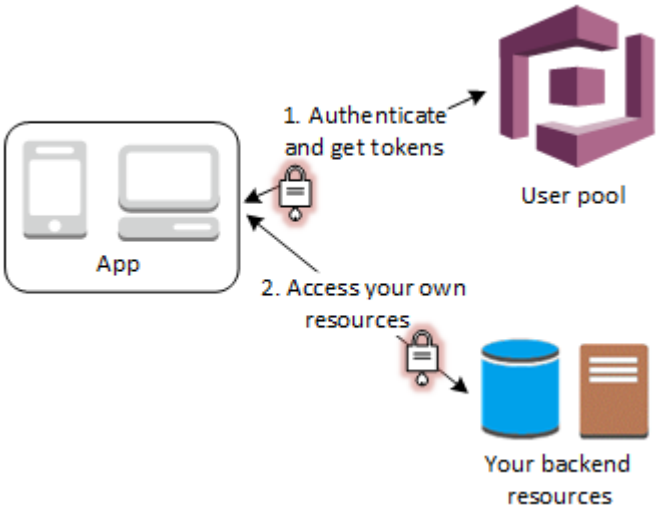
成功进行身份验证后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌来检索允许您的应用程序访问其他 Amazon 服务的 Amazon 证书，也可以选择使用它们来控制对服务器端资源或 Amazon API Gateway 的访问权限。

有关更多信息，请参阅身份验证会话示例 和了解用户池 JSON 网络令牌 (JWTs)。



使用用户池令牌访问后端资源

成功进行用户池登录后，您的 Web 或移动应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用这些令牌控制对您的服务器端资源的访问。您也可以创建用户池组来管理权限以及表示不同类型的用户。有关使用组控制资源访问权限的更多信息，请参阅向用户池添加组。



在为用户群体配置域后，Amazon Cognito 会预调配一个托管 Web UI，您可使用此 UI 为应用程序添加注册页和登录页。使用此 OAuth 2.0 基础，您可以创建自己的资源服务器，让您的用户能够访问受保护的资源。有关更多信息，请参阅 作用域、M2M 以及 APIs 带资源服务器的场景。

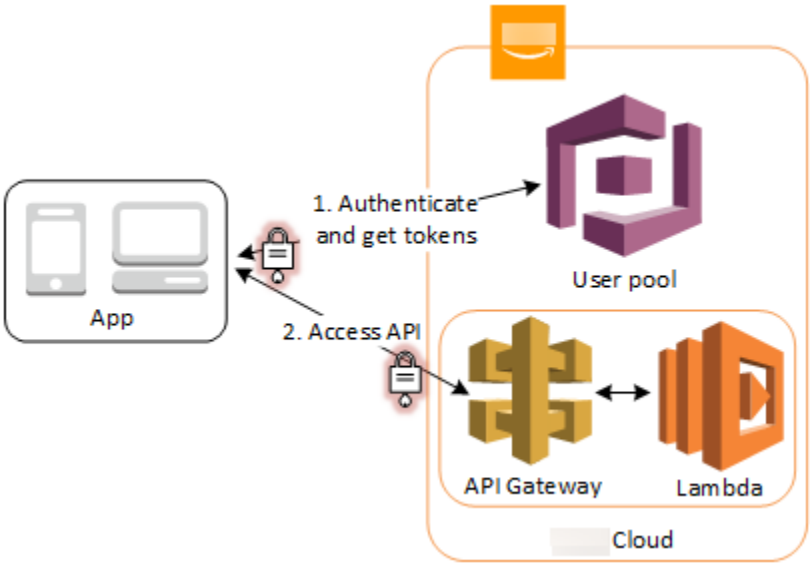
有关用户群体身份验证的更多信息，请参阅身份验证会话示例和了解用户池 JSON 网络令牌 (JWTs)。

将 API Gateway 和 Lambda 与用户池结合使用来访问资源

您可以允许用户通过 API Gateway 访问您的 API。API Gateway 会验证来自成功用户池身份验证的令牌，并使用它们向您的用户授予对资源（包括 Lambda 函数）或您自己的 API 的访问权限。

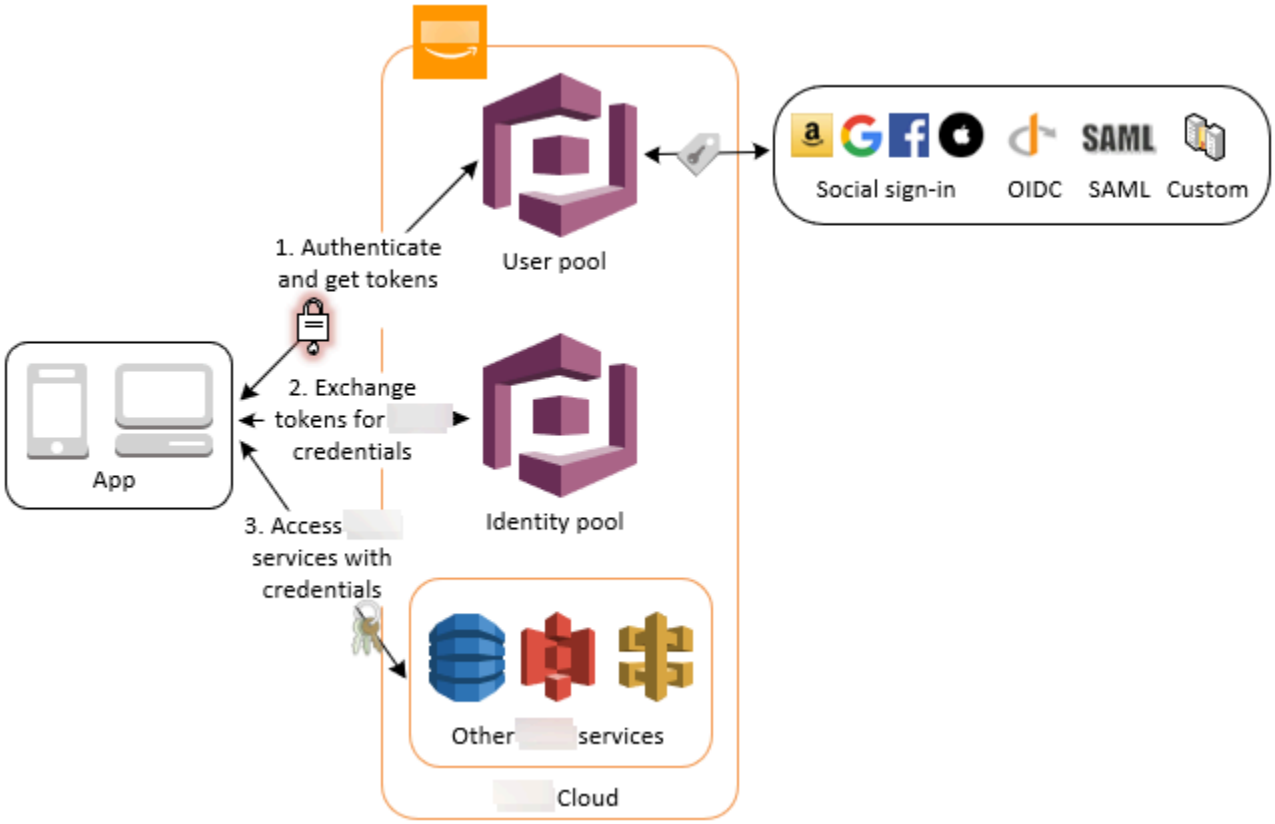
您可以使用用户池中的组控制对 API Gateway 的权限，方法是将组成员资格映射到 IAM 角色。用户所属的组包含在您的应用程序用户登录时用户池提供的 ID 令牌中。有关用户池组的更多信息，请参阅向用户池添加组。

您可以将您的用户池令牌随请求一起提交到 API Gateway，以便 Amazon Cognito 授权方 Lambda 函数进行验证。有关 API Gateway 的更多信息，请参阅将 API Gateway 与 Amazon Cognito 用户池结合使用。



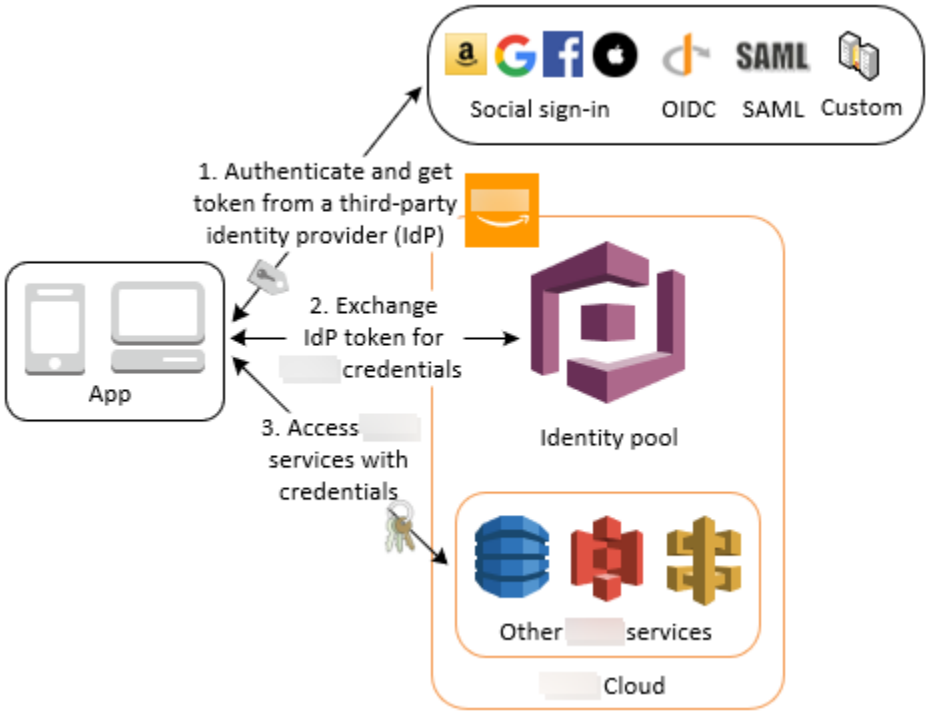
使用用户池和身份池访问 Amazon 服务

成功进行用户池身份验证后，您的应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以将它们交换为通过身份池临时访问其他 Amazon 服务。有关更多信息，请参阅在登录后使用身份池访问 Amazon Web Services 服务 和Amazon Cognito 身份池入门。



借助第三方进行身份验证并使用身份池访问 Amazon 服务

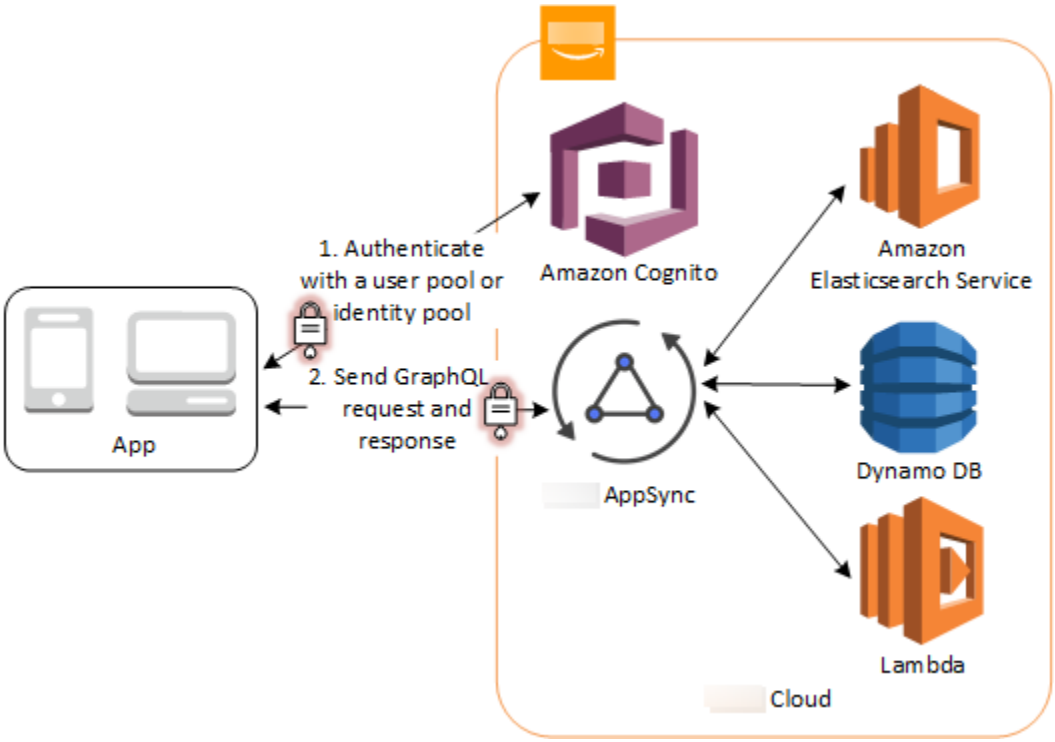
您可以允许您的用户通过身份池访问 Amazon 服务。身份池需要来自由第三方身份提供商进行身份验证的用户的 IdP 令牌 (如果是匿名来访者，则不需要令牌)。作为交换，身份池会授予可用于访问其他 Amazon 服务的临时 Amazon 证书。有关更多信息，请参阅 Amazon Cognito 身份池入门。



使用 Amazon Cognito 访问 Amazon AppSync 资源

您可以通过成功的 Amazon Cognito 用户池身份验证获得的令牌向您的用户授予访问 Amazon AppSync 资源的权限。有关更多信息，请参阅《Amazon AppSync 开发者指南》中的AMAZON_COGNITO_USER_POOLS 授权。

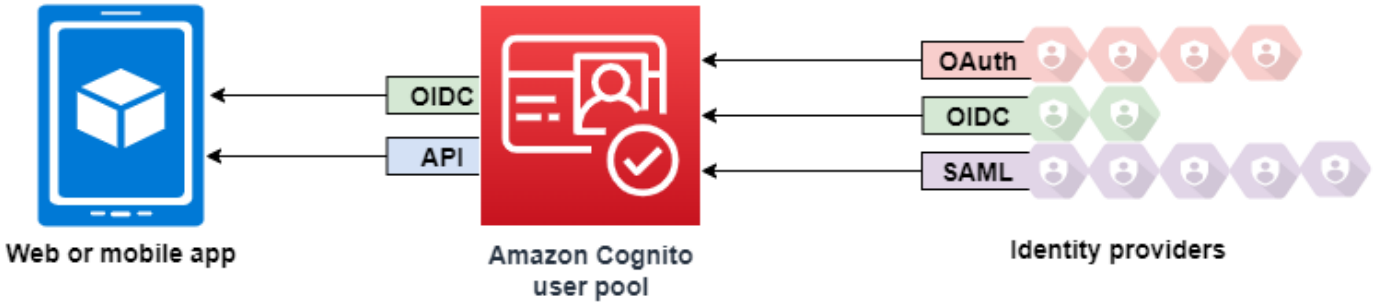
您还可以使用从身份池收到的 IAM 证书签署对 Amazon AppSync GraphQL API 的请求。参见AWS_IAM授权。


1. Authenticate
with a user pool or
identity pool
Amazon Cognito
Amazon
Elasticsearch Service
App
2. Send GraphQL
request and
response
AppSync
Dynamo DB
Lambda
Cloud

Amazon Cognito 用户群体

Amazon Cognito 用户群体是用于 Web 和移动应用程序身份验证和授权的用户目录。从应用程序的角度来看，Amazon Cognito 用户群体是 OpenID Connect（OIDC）身份提供者（IdP）。用户群体为安全性、身份联合验证、应用程序集成和用户体验自定义添加了多层附加功能。

例如，您可以验证用户的会话是否来自可信来源。您可以将 Amazon Cognito 目录与外部身份提供者相结合。使用您的首选 Amazon SDK，您可以选择最适合您的应用程序的 API 授权模式。您还可以添加用于修改或彻底修改 Amazon Cognito 原定设置行为的 Amazon Lambda 函数。



主题

- 特征
- 用户池功能计划
- Amazon Cognito 用户池安全最佳实践
- 使用 Amazon Cognito 用户池进行身份验证
- 用户池使用第三方身份提供商登录
- 用户池托管登录
- 使用 Lambda 触发器自定义用户池工作流
- 管理用户池中的用户
- 了解用户池 JSON 网络令牌 (JWTs)
- 在成功登录后访问资源
- 配置用户池功能
- 使用 Amazon Cognito 用户池安全功能
- 用户池端点和托管登录参考

特征

Amazon Cognito 用户群体具有以下功能。

注册

Amazon Cognito 用户群体具有用户驱动、管理员驱动和编程方法，可将用户配置文件添加到您的用户群体。Amazon Cognito 用户群体支持以下注册模式。您可以在应用程序中使用这些模型的任意组合。

> **Important**
>
> 如果您在用户群体中激活用户注册，则互联网上的任何人都可以注册账户并登录您的应用程序。除非您开放您的应用程序供公开注册，否则不要在用户群体中启用自助注册。要更改此设置，请在用户池控制台的 “身份验证” 下的 “注册” 菜单中更新自助服务注册，或者更新 CreateUserPool或 UpdateUserPoolAPI AllowAdminCreateUserOnly请求中的值。
> 有关可以在用户群体中设置的安全特征的信息，请参阅使用 Amazon Cognito 用户池安全功能。

1. 用户可以在应用程序中输入其信息，并创建用户群体原生的用户配置文件。您可以调用 API 注册操作在用户群体中注册用户。您可以向任何人开放这些注册操作，也可以使用客户密钥或 Amazon 凭据对其进行授权。
2. 您可以将用户重定向到第三方 IdP，他们可以授权该第三方 IdP 将其信息传递给 Amazon Cognito。Amazon Cognito 将 OIDC 身份令牌、 OAuth 2.0 `userInfo` 数据和 SAML 2.0 断言处理到您的用户池中的用户个人资料中。您可以根据属性映射规则控制您希望 Amazon Cognito 接收的属性。
3. 您可以跳过公共注册或联合身份验证注册，并根据自己的数据来源和模式来创建用户。在 Amazon Cognito 控制台或 API 中直接添加用户。从 CSV 文件导入用户。运行一个 just-in-time Amazon Lambda 函数，在现有目录中查找您的新用户，并根据现有数据填充他们的用户配置文件。

用户注册后，您可以将他们添加到 Amazon Cognito 在访问令牌和 ID 令牌中列出的组。在将 ID 令牌传递给身份池时，还可以将用户群体组链接到 IAM 角色。

相关 主题

- 管理用户池中的用户
- 了解 API、OIDC 和托管登录页面身份验证

- 使用 Amazon Cognito 身份提供商的代码示例 Amazon SDKs

登录

Amazon Cognito 可以是应用程序的独立用户目录和身份提供者（IdP）。您的用户可以使用由 Amazon Cognito 托管的托管登录页面登录，也可以通过 Amazon Cognito 用户池 API 使用定制的用户身份验证服务登录。自定义前端背后的应用程序层可以使用几种方法中的任何一种来授权后端的请求，以确认合法请求。

用户可以使用用户名和密码、密钥以及电子邮件和短信一次性密码设置和签名。您可以提供使用外部用户目录进行合并登录、登录后的多重身份验证 (MFA)、信任记住的设备以及您设计的自定义身份验证流程。

要使用外部目录（可选择与 Amazon Cognito 内置的用户目录结合使用）登录用户，您可以添加以下集成。

1. 使用 OAuth 2.0 社交登录登录并导入客户用户数据。亚马逊 Cognito 支持在 2.0 之前使用谷歌、Facebook、亚马逊和苹果登录。 OAuth
2. 使用 SAML 和 OIDC 登录登录并导入工作和学校用户数据。您也可以将 Amazon Cognito 配置为接受来自任何 SAML 或 OpenID Connect（OIDC）身份提供者（IdP）的声明。
3. 将外部用户配置文件链接到原生用户配置文件。关联的用户可以使用第三方用户身份登录，并获得您分配给内置目录中的用户的访问权限。

相关 主题

- 用户池使用第三方身份提供商登录
- 将联合用户与现有用户配置文件关联

Machine-to-machine 授权

有些会话不是 human-to-machine互动。您可能需要一个能够通过自动化流程向 API 授权请求的服务账户。要生成访问令牌以 machine-to-machine进行 OAuth 2.0 范围的授权，您可以添加生成客户端凭据授权的应用程序客户端。

相关 主题

- 作用域、M2M 以及 APIs 带资源服务器的场景

托管登录

当您不想构建用户界面时，可以向用户展示自定义的托管登录页面。托管登录是一组用于注册、登录、多因素身份验证 (MFA) 和密码重置的网页。您可以将托管登录添加到现有域中，也可以在 Amazon 子域中使用前缀标识符。

相关 主题

- 用户池托管登录
- 配置用户池域

安全性

您的本地用户可以使用来自短信或电子邮件的代码，或者生成多因素身份验证 (MFA) 代码的应用程序来提供额外的身份验证因子。您可以构建用于在应用程序中设置和处理 MFA 的机制，也可以让托管登录对其进行管理。当您的用户从可信设备登录时，Amazon Cognito 用户群体可以绕过 MFA。

如果您不想最初就要求用户提供 MFA，则可以有条件地提出要求。借助高级安全功能，Amazon Cognito 可以检测潜在的恶意活动，并要求用户设置 MFA 或阻止登录。

如果进入用户池的网络流量可能是恶意的，则可以对其进行监控并使用 Amazon WAF 网络采取措施 ACLs。

相关 主题

- 向用户池添加 MFA
- 具有威胁防护功能的高级安全性
- 将 Amazon WAF Web ACL 与用户池关联

自定义客户体验

在用户注册、登录或配置文件更新的大多数阶段，您可以自定义 Amazon Cognito 处理请求的方式。使用 Lambda 触发器，您可以根据自定义条件修改 ID 令牌或拒绝注册请求。您可以创建自己的自定义身份验证流程。

您可以上传自定义 CSS 和徽标，为用户提供熟悉的托管登录外观和感觉。

相关 主题

- 使用 Lambda 触发器自定义用户池工作流
- 自定义身份验证质询 Lambda 触发器
- 将品牌应用于托管登录页面

监控和分析

Amazon Cognito 用户池将 API 请求（包括托管登录请求）记录到。 Amazon CloudTrail您可以在 Service Quotas 控制台中查看亚马逊 CloudWatch 日志中的性能指标， CloudWatch 使用 Lambda 触发器将自定义日志推送到，监控电子邮件和短信的传送情况，以及监控 API 请求量。

借助 Plus 功能计划，您可以使用自动学习技术监控用户身份验证尝试，以确定是否存在漏洞迹象，并立即修复风险。这些高级安全功能还可以将用户活动记录到您的用户池中，也可以记录到 Amazon S3、 CloudWatch 日志或 Amazon Data Firehose 中。

还可以将 API 请求中的设备和会话数据记录到 Amazon Pinpoint 活动中。借助 Amazon Pinpoint，您可以根据对用户活动的分析，从应用程序发送推送通知。

相关 主题

- 亚马逊 Cognito 正在登录 Amazon CloudTrail
- 跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况
- 从 Amazon Cognito 用户池导出日志
- 使用 Amazon Pinpoint 进行用户池分析

Amazon Cognito 身份池集成

Amazon Cognito 的另一半是身份池。身份池提供证书，用于授权和监控您的用户向（例如 Amazon DynamoDB 或 Amazon S3）发出的 API 请求。 Amazon Web Services 服务您可以构建基于身份的访问策略，根据您在用户群体中对用户进行分类的方式来保护您的数据。身份池还可以接受来自各种身份提供者的令牌和 SAML 2.0 断言，与用户群体身份验证无关。

相关 主题

- 在登录后使用身份池访问 Amazon Web Services 服务
- Amazon Cognito 身份池

用户池功能计划

了解成本是准备实施 Amazon Cognito 用户池身份验证的关键步骤。Amazon Cognito 有针对用户池的功能计划。每个计划都有一组功能和每位活跃用户的月度费用。每个功能计划都比之前的功能计划解锁了更多功能的访问权限。

用户池具有多种功能，您可以打开和关闭这些功能。例如，您可以开启多重身份验证 (MFA) 和关闭使用第三方身份提供商登录 ()。IdPs有些更改需要您切换功能计划。用户池的以下特征决定了每月向您 Amazon 收取的使用费用。

- 您选择的功能
- 您的应用程序每秒向用户池 API 发出的请求
- 一个月内有身份验证、更新或查询活动的用户数，也称为每月活跃用户或 MAUs
- 来自第三方 SAML 2.0 或 OpenID Connect (OIDC) 的每月活跃用户数 IdPs
- 使用客户端凭证授予授权的应用程序客户端和用户池的 machine-to-machine数量

有关用户池定价的最新信息，请参阅 Amazon Cognito 定价。

功能计划选项适用于一个用户池。同一个用户池中的不同用户池 Amazon Web Services 账户 可以有不同的计划选择。您不能将单独的功能计划应用于用户池中的应用程序客户端。新用户池的默认计划选择是 “基本套餐”。

您可以随时在功能计划之间切换，以满足应用程序的要求。计划之间的某些更改要求您关闭活动功能。有关更多信息，请参阅 关闭功能以更改功能计划。

用户池功能计划

精简版

Lite 是一项低成本的功能计划，适用于每月活跃用户数量较少的用户群。对于具有基本身份验证功能的用户目录，此计划已经足够了。它包括登录功能和经典的托管用户界面，这是托管登录的更精简、可定制性更低的前身。许多较新的功能，例如访问令牌自定义和密钥身份验证，都未包含在精简版计划中。

必需品

Essentials 具有所有最新的用户池身份验证功能。该计划为您的应用程序添加了新的选项，无论您的登录页面是托管登录还是自定义登录。Essentials 具有高级身份验证功能，例如基于选择的登录和电子邮件 M FA。

再加上

Plus 包含 Essentials 计划中的所有内容，并添加了保护用户的高级安全功能。监控用户登录、注册和密码管理请求，寻找泄露迹象。例如，用户池可以检测用户是从意想不到的位置登录，还是使用了属于公共漏洞的密码。

使用 Plus 套餐的用户池会生成用户活动详细信息和风险评估日志。将这些日志导出到外部服务时，您可以对这些日志进行自己的使用和安全分析。

> **Note**
>
> 以前，一些用户池功能包含在高级安全功能定价结构中。此结构中包含的功能现在属于Essentials或Plus计划。

主题

- 选择功能计划
- 按计划划分的功能
- 基本计划功能
- 加上套餐功能
- 关闭功能以更改功能计划

选择功能计划

Amazon Web Services Management Console

选择功能计划

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “设置” 菜单并查看 “功能计划” 选项卡。
5. 查看精简版、Essentials 和 Plus 套餐中可供你使用的功能。
6. 要更改套餐，请选择 “切换到基本套餐” 或 “切换到高级版”。要切换到精简版套餐，请选择其他套餐，然后选择与精简版比较。

7. 在下一个屏幕上，查看您的选择并选择确认。

CLI/API/SDK

CreateUserPool和UpdateUserPool操作在`UserPoolTier`参数中设置您的功能计划。如果未为指定值`UserPoolTier`，则您的用户池默认为`Essentials`。如果设置`AdvancedSecurityMode`为`AUDIT`或`ENFORCED`，则用户池等级必须为`PLUS`，`PLUS`如果未指定，则默认为。

有关语法，CreateUserPool请参阅中的示例。 CreateUserPool有关各种编程语言中此函数的链接，Amazon SDKs 请参阅 of 中的。

```
"UserPoolTier": "PLUS"
```

在中 Amazon CLI，此选项是`--user-pool-tier`参数。

```
--user-pool-tier PLUS
```

有关更多信息 create-user-pool，请参阅 Amazon CLI 命令参考update-user-pool中的和。

按计划划分的功能

用户池中的功能和计划

特征	描述	功能计划
防范不安全的密码	在运行时检查纯文本密码是否有泄露迹象	再加上
防范恶意登录尝试	在运行时检查会话属性是否存在漏洞迹象	再加上
记录和分析用户活动	生成用户身份验证会话属性和风险评分的日志	再加上
导出用户活动日志	将用户会话和风险日志推送到外部 Amazon Web Services 服务	再加上

特征	描述	功能计划
使用可视化编辑器自定义托管登录页面	使用 Amazon Cognito 控制台中的可视化编辑器将品牌和风格应用于您的托管登录页面	基本款 + Plus
带有电子邮件一次性验证码的 MFA	请求或要求本地用户在用户名身份验证后提供额外的电子邮件登录系数	基本款 + Plus
在运行时自定义访问令牌范围和声明	使用 Lambda 触发器扩展用户池访问令牌的授权功能	基本款 + Plus
使用一次性验证码进行无密码登录	允许用户通过电子邮件或短信接收一次性密码，作为他们的第一个身份验证因素	基本款 + Plus
使用硬件或软件身份验证器进行密钥登录 FIDO2	允许用户使用存储在 FIDO2 身份验证器上的加密密钥作为他们的第一个身份验证因素	基本款 + Plus
注册和登录	执行身份验证操作并允许新用户在您的应用程序中注册帐户。	精简版 + 基础版 + Plus
用户组	创建用户的逻辑分组并为身份池操作分配默认 IAM 角色。	精简版 + 基础版 + Plus
使用社交、SAML 和 OIDC 提供商登录	为用户提供直接登录或使用其首选提供商登录的选项。	精简版 + 基础版 + Plus
OAuth 2.0/OIDC 授权服务器	担任 OIDC 发行人。	精简版 + 基础版 + Plus
登录页面	用于身份验证的网页托管集合。基础版和高级版提供托管登录。所有功能层均提供经典托管用户界面。	精简版 + 基础版 + Plus

特征	描述	功能计划
密码、自定义、刷新令牌和 SRP 身份验证	在您的应用程序中提示用户输入用户名和密码。	精简版 + 基础版 + Plus
Machine-to-machine (M2M) 带有客户凭证	发放访问令牌以授权非人类实体。	精简版 + 基础版 + Plus
使用资源服务器进行 API 授权	颁发具有自定义范围的访问令牌，以授权对外部系统的访问。	精简版 + 基础版 + Plus
用户导入	设置从 CSV 文件导入任务，并在用户登录时对其进行 just-in-time迁移。	精简版 + 基础版 + Plus
带有身份验证器应用程序和 SMS 一次性代码的 MFA	请求或要求本地用户在用户名身份验证后提供额外的 SMS 消息或身份验证器应用程序登录因子	精简版 + 基础版 + Plus
在运行时自定义 ID 令牌范围和声明	使用 Lambda 触发器扩展用户池身份 (ID) 令牌的身份验证功能	精简版 + 基础版 + Plus
使用 Lambda 触发器的自定义运行时操作	使用执行外部操作和影响身份验证的 Lambda 函数在运行时自定义登录流程	精简版 + 基础版 + Plus
使用 CSS 自定义托管登录页面	下载 CSS 模板并更改托管登录页面中的一些样式	精简版 + 基础版 + Plus

基本计划功能

Essentials 功能计划包含 Amazon Cognito 用户池中的大部分最佳和最新功能。当你从 Lite 套餐切换到 Essentials 套餐时，你将获得托管登录页面的新功能、使用电子邮件消息一次性密码的多因素身份验证、增强的密码策略和自定义访问令牌。要继续 up-to-date使用新的用户池功能，请为您的用户池选择 Essentials 套餐。

以下各节简要概述了您可以通过 Essentials 计划向应用程序添加的功能。有关详细信息，请参阅以下页面。

其他资源

- 访问令牌自定义：令牌生成前 Lambda 触发器
- 向 MFA 发送电子邮件： 短信和电子邮件消息 MFA
- 密码历史记录：密码、账户恢复和密码策略
- 增强的用户界面：将品牌应用于托管登录页面

主题

- 访问令牌自定义
- 电子邮件 MFA
- 防止密码重用
- 托管登录托管登录和授权服务器
- 基于选择的身份验证

访问令牌自定义

用户池访问令牌向应用程序授予以下权限：访问 API、从 userInfo 端点检索用户属性或为外部系统建立组成员关系。在高级场景中，您可能需要将应用程序在运行时确定的其他临时参数添加到用户池目录中的默认访问令牌数据中。例如，您可能需要使用 Amazon Verified Permissions 验证用户的 API 权限，并相应地调整访问令牌中的范围。

Essentials计划增加了代币生成前触发器的现有功能。对于较低级别的套餐，您可以使用额外的声明、角色和群组成员资格来自定义 ID 令牌。Essentials 添加了触发器输入事件的新版本，用于自定义访问令牌声明、角色、群组成员资格和范围。在事件版本三中， machine-to-machine（M2M）客户端凭证授予可自定义访问令牌。

自定义访问令牌

1. 选择基本版或高级版功能计划。
2. 为触发器创建 Lambda 函数。要使用我们的示例函数，请为其配置 Node.js。
3. 使用我们的示例代码填充您的 Lambda 函数，或者自己编写函数。您的函数必须处理来自 Amazon Cognito 的请求对象，并返回您想要包括的更改。

4. 将您的新函数指定为第二版或第三版的代币生成前触发器。第二版事件自定义用户身份的访问令牌。第三版自定义了用户和计算机身份的访问令牌。

了解更多

- 自定义访问令牌
- 如何在 Amazon Cognito 用户池中自定义访问令牌

电子邮件 MFA

可以将 Amazon Cognito 用户池配置为使用电子邮件作为多因素身份验证 (MFA) 的第二个因素。通过电子邮件 MFA，Amazon Cognito 可以向用户发送一封包含验证码的电子邮件，用户必须输入该验证码才能完成身份验证过程。这样就为用户登录流程额外增加了一层重要的安全保护。要启用基于电子邮件的 MFA，必须将用户池配置为使用 Amazon SES 电子邮件发送配置，而不是默认的电子邮件配置。

当您的用户选择通过电子邮件进行 MFA 时，每当用户尝试登录时，Amazon Cognito 都会向其注册的电子邮件地址发送一次性验证码。然后，用户必须将此验证码提供给用户池，这样才能完成身份验证流程并获得访问权限。这样可以确保即使用户的用户名和密码遭到泄露，他们还必须提供额外的因素（通过电子邮件发送的验证码），然后才能访问您的应用程序资源。

有关更多信息，请参阅 短信和电子邮件消息 MFA。下面概述了如何设置用户池和用户以使用电子邮件 MFA。

在亚马逊 Cognito 控制台中设置电子邮件 MFA

1. 选择基本版或高级版功能计划。
2. 在用户池的登录菜单中，编辑多重身份验证。
3. 选择要设置的 MFA 强制执行级别。使用需要 MFA 选项时，API 中的用户会自动收到质询，要求他们使用 MFA 进行设置、确认和登录。在需要 MFA 的用户池中，托管登录会提示他们选择和设置 MFA 因子。使用可选 MFA 选项时，您的应用程序必须为用户提供选项来设置 MFA 和设置用户电子邮件 MFA 首选项。
4. 在 MFA 方法下，选择电子邮件消息作为其中一个选项。

了解更多

- 短信和电子邮件消息 MFA

防止密码重用

默认情况下，Amazon Cognito 用户池密码策略会设置密码长度和字符类型要求以及临时密码到期时间。Essentials 计划增加了强制执行密码历史记录的功能。当用户尝试重置其密码时，您的用户池会阻止用户将其设置为以前的密码。有关配置密码策略的更多信息，请参阅添加用户池密码要求。下面概述了如何使用密码历史记录策略来设置用户池。

在 Amazon Cognito 控制台中设置密码历史记录

1. 选择基本版或高级版功能计划。
2. 在用户池的 “身份验证方法” 菜单中，找到 “密码策略”，然后选择 “编辑”。
3. 配置其他可用选项，并为防止使用之前的密码设置一个值。

了解更多

- 密码、账户恢复和密码策略

托管登录托管登录和授权服务器

Amazon Cognito 用户池具有支持以下功能的可选网页：OpenID Connect (OIDC) IdP、 IdPs第三方的服务提供商或依赖方，以及用于注册和登录的公共用户交互页面。这些页面统称为托管登录。当您为用户池选择域名时，Amazon Cognito 会自动激活这些页面。精简版套餐包含托管用户界面，而 Essentials 套餐则会打开这个高级版本的注册和登录页面。

托管登录页面具有简洁的 up-to-date界面，具有更多用于自定义品牌和样式的功能和选项。Essentials 计划是解锁托管登录访问权限的最低套餐级别。

在 Amazon Cognito 控制台中设置托管登录

1. 从 “设置” 菜单中，选择 “必备” 或 “增强” 功能计划。
2. 在域名菜单中，将域名分配给您的用户池，然后选择托管登录的品牌化版本。
3. 在托管登录菜单的样式选项卡下，选择创建样式并将样式分配给应用程序客户端，或者创建新的应用程序客户端。

了解更多

- 用户池托管登录

基于选择的身份验证

Essentials 层为增强型 UI 中的身份验证操作和基于 SDK 的 API 操作引入了新的身份验证流程。此流程是基于选择的身份验证。基于选择的身份验证是一种方法，在这种方法中，用户的身份验证不是从应用程序端的登录方法声明开始，而是从查询可能的登录方法开始，然后进行选择。您可以将用户池配置为支持基于选择的身份验证和解锁用户名-密码、无密码和密钥身份验证。在 API 中，USER_AUTH流程就是这样。

在 Amazon Cognito 控制台中设置基于选择的身份验证

1. 选择基本版或高级版功能计划。
2. 在用户池的登录菜单中，编辑基于选择的登录选项。选择并配置要在基于选择的身份验证中启用的身份验证方法。
3. 在用户池的身份验证方法菜单中，编辑登录操作的配置。

了解更多

- 使用 Amazon Cognito 用户池进行身份验证

加上套餐功能

Plus 功能计划为 Amazon Cognito 用户池提供了高级安全功能。这些功能在运行时记录和分析用户上下文，以发现设备、位置、请求数据和密码中的潜在安全问题。然后，他们会通过自动响应来缓解潜在风险，从而屏蔽或为用户帐户添加安全保护措施。您也可以将安全日志导出到 Amazon S3、Amazon Data Firehose 或亚马逊 CloudWatch 日志以供进一步分析。

当你从 Essentials 切换到 Plus 套餐时，你将获得 Essentials 中的所有功能以及随之而来的其他功能。其中包括威胁防护安全选项集，也称为高级安全功能。要将您的用户池配置为自动适应身份验证前端中的威胁，请为您的用户池选择 Plus 套餐。

以下各节简要概述了您可以通过 Plus 计划向应用程序添加的功能。有关详细信息，请参阅以下页面。

其他资源

- 自适应身份验证：使用自适应身份验证
- 凭据泄露：使用已泄露的凭证检测
- 日志导出：从 Amazon Cognito 用户池导出日志

主题

- 威胁防护：自适应身份验证
- 威胁防护：凭据泄露检测
- 威胁防护：用户活动记录

威胁防护：自适应身份验证

Plus 计划包括自适应身份验证功能。激活此功能后，您的用户池将对每个用户身份验证会话进行风险评估。根据生成的风险评级，对于风险等级高于您确定的阈值的用户，您可以屏蔽身份验证或推送 MFA。使用自适应身份验证，您的用户池和应用程序会自动为您怀疑其账户受到攻击的用户屏蔽或设置 MFA。您还可以就用户群中的风险评级提供反馈，以调整未来的评级。

在 Amazon Cognito 控制台中设置自适应身份验证

1. 选择 Plus 功能计划。
2. 在用户池的威胁防护菜单中，编辑威胁防护下的标准和自定义身份验证。
3. 将标准或自定义身份验证的强制模式设置为全功能。
4. 在 “自适应身份验证” 下，为不同风险级别配置自动风险响应。

了解更多

- 使用自适应身份验证
- 在应用程序中收集威胁防护的数据

威胁防护：凭据泄露检测

Plus 计划包括凭据泄露检测功能。此功能可防止使用不安全的密码以及这种做法造成的意外访问应用程序的威胁。当您允许用户使用用户名和密码登录时，他们可能会重复使用他们在其他地方使用的密码。该密码可能已被泄露，或者只是人们经常猜到的。通过凭证泄露检测，您的用户池可以读取用户提交的密码，并将其与密码数据库进行比较。如果该操作导致决定密码可能被泄露，则可以将用户池配置为阻止登录，然后在应用程序中为该用户启动密码重置。

在新用户注册、现有用户登录以及用户尝试重置密码时，凭证受损检测可能会对不安全的密码做出反应。使用此功能，无论用户在何处输入不安全的密码，您的用户池都可以阻止或警告使用不安全的密码登录。

在 Amazon Cognito 控制台中设置凭证泄露检测

1. 选择 Plus 功能计划。
2. 在用户池的威胁防护菜单中，编辑威胁防护下的标准和自定义身份验证。
3. 将标准或自定义身份验证的强制模式设置为全功能。
4. 在 “已泄露的凭据” 下，配置要检查的身份验证操作的类型，以及要从用户池中获得的自动响应。

了解更多

- 使用已泄露的凭证检测

威胁防护：用户活动记录

Plus 计划增加了日志记录功能，可提供安全分析和用户身份验证尝试的详细信息。您可以查看风险评估、用户 IP 地址、用户代理以及有关连接到您的应用程序的设备的其他信息。您可以使用内置的威胁防护功能对这些信息采取行动，也可以分析自己系统中的日志并采取适当的措施。您可以将威胁防护中的日志导出到 Amazon S3、 CloudWatch 日志或 Amazon DynamoDB。

在 Amazon Cognito 控制台中设置用户活动日志

1. 选择 Plus 功能计划。
2. 在用户池的威胁防护菜单中，编辑威胁防护下的标准和自定义身份验证。
3. 将标准或自定义身份验证的强制模式设置为仅限审计。这是日志的最低设置。您也可以在全功能模式下将其激活，并配置其他威胁防护功能。
4. 要将日志导出到其他日志以 Amazon Web Services 服务 供第三方分析，请转到用户池的 “日志流” 菜单并设置导出目的地。

了解更多

- 导出用户身份验证事件
- 从 Amazon Cognito 用户池导出日志

关闭功能以更改功能计划

功能计划为您的用户池添加配置选项。只有在相关功能计划处于活动状态时，您才能配置和使用这些功能。例如，您可以在 Plus 和 Essentials 计划中配置访问令牌自定义，但不能在 Lite 计划中配置访问令

牌自定义。要停用这些特征，必须停用每个活动组件。Amazon Cognito 控制台的 “设置” 菜单中的 “切换到” 选项会通知您必须先停用哪些功能，然后才能更改功能计划。在本章中，您可以了解停用特征对用户池配置所作的更改，以及如何单独关闭这些特征。

访问令牌自定义

要切换到不包括访问令牌自定义的计划，您必须从用户池中移除令牌生成前 Lambda 触发器。要在不自定义访问令牌的情况下添加新的令牌生成前触发器，请为该触发器分配新函数并对 `V1_0` 事件进行配置。这些版本一触发器事件只能处理 ID 令牌的更改。

要手动停用访问令牌自定义，请移除令牌生成前触发器，然后添加一个新的版本一触发器。

威胁防护

要切换到没有威胁防护的计划，请停用用户池的威胁防护菜单中的所有功能。

日志导出

要切换到不导出日志的计划，请从用户池的 “日志流” 菜单中将其停用。您的用户池不再生成本地或导出的用户活动日志。您也可以发送 SetLogDeliveryConfigurationAPI 请求，删除任何`EventSource`值为的配置`UserActivity`。

电子邮件 MFA

要切换到不使用电子邮件 MFA 的套餐，请前往用户池的登录菜单。编辑多重身份验证并取消选择电子邮件作为可用的 MFA 方法之一。

Amazon Cognito 用户池安全最佳实践

本页介绍了当你想防范常见威胁时可以实施的安全最佳实践。您选择的配置将取决于每个应用程序的用例。我们建议您至少对管理操作应用最低权限，并采取措施保护应用程序和用户密钥。您可以采取的另一个高级但有效的步骤是配置 Amazon WAF Web 并将其应用 ACLs 于您的用户池。

在网络层面保护您的用户池

Amazon WAF web ACLs 可以保护您使用 Amazon Cognito 构建的身份验证机制的性能和成本。借助 Web ACLs，您可以在 API 和托管登录请求之前设置防护栏。Web ACLs 创建网络层和应用层过滤器，这些过滤器可以根据您设计的规则丢弃流量或要求使用验证码。请求只有在满足您的 Web ACL 规则中的条件后才会传递到您的 Amazon Cognito 资源。有关更多信息，请参阅 Amazon WAF web ACLs。

防范短信滥用

如果您允许在用户池中进行公开注册，则可以使用 Amazon Cognito 在 SMS 短信中发送的代码来配置账户验证。SMS 消息可能与不想要的活动相关联，从而增加您的 Amazon 账单。配置您的基础架构，使其能够抵御在欺诈情况下发送 SMS 消息。有关更多信息，请查看 Amazon 博客中的以下帖子。

- 使用 Amazon Cognito 用户池降低用户注册欺诈和短信激增的风险
- 防范短信抽水：新 Amazon 功能可帮助抵御人为膨胀的流量

了解公共身份验证

Amazon Cognito 用户池具有客户身份和访问管理 (CIAM) 功能，这些功能支持公众可以注册用户账户并访问您的应用程序的用例。当用户池允许自助注册时，它可以接受来自公共互联网的用户帐户请求。自助服务请求来自和之类的 API 操作 InitiateAuth，SignUp以及用户与托管登录的交互。您可以配置用户池以减少可能来自公共请求的滥用行为，或者完全禁用公共身份验证操作。

以下设置是您可以在用户池和应用程序客户端中管理公共和内部身份验证请求的一些方法。

影响公共用户池访问权限的用户池设置示例

设置	可用选项	已配置于	对公共身份验证的影响	控制台设置	API 操作和参数
自助注册	允许用户以管理员身份注册帐户或创建用户帐户。	用户群体	阻止公开注册	注册-自助注册	CreateUserPool, UpdateUserPool `AdminCreateUserConfig – AllowAdminCreateUserOnly`
管理员确认	向新用户发送确认码或要求管理员进行确认。	用户群体	防止在没有管理员操作的情况下确认注册	注册 — Cognito 辅助的验证和确认	CreateUserPool, UpdateUserPool

设置	可用选项	已配置于	对公共身份验证的影响	控制台设置	API 操作和参数
					AccountRecoverySettings_admin_only
用户披露	在登录和密码重置时发送“未找到用户”消息，或者防止泄露。	用户群体	防止猜测登录名、电子邮件地址或电话号码	应用程序客户端-防止用户存在错误	CreateUserPoolClient, UpdateUserPoolClient PreventUserExistenceErrors
客户端密钥	在注册、登录、密码重置时需要或不需要密钥哈希	应用程序客户端	防范来自未经授权来源的身份验证请求	应用程序客户端-客户端密钥	CreateUserPoolClient GenerateSecret
Web ACLs	为身份验证请求启用或不启用网络防火墙	用户群体	根据管理员定义的请求特征和 IP 地址规则限制或阻止访问	Amazon WAF—WAF 设置	AssociateWebACL ResourceArn

设置	可用选项	已配置于	对公共身份验证的影响	控制台设置	API 操作和参数
外部 IdP	允许用户在第三方 IdPs、用户池目录或两者中登录	应用程序客户端	将本地用户或联合用户排除在注册和登录之外。	应用程序客户端-身份提供商	CreateUserPoolClient, UpdateUserPoolClient `SupportedIdentityProviders`
授权服务器	托管或不托管公共网页进行身份验证	用户群体	关闭公共网页，只允许基于 SDK 的身份验证	域	CreateUserPoolDomain 创建任何用户池域都会使公共网页可用。
威胁防护	启用或禁用对恶意活动迹象或不安全密码的监控	用户池或应用程序客户端	当用户显示泄露迹象时，可以自动阻止登录或要求 MFA	威胁防护-防护设置	SetRiskConfiguration 的参数`SetRiskConfiguration` 定义您的威胁防护设置。

使用客户机密保护机密客户

客户端密钥是与应用程序客户端关联的可选字符串。向具有客户端密钥的应用程序客户端发出的所有身份验证请求都必须包含由用户名、客户端 ID 和客户端密钥生成的密钥哈希。那些不知道客户机密的人从一开始就被拒之门外。

但是，客户机密有局限性。如果您在公共客户端软件中嵌入了客户端密钥，则您的客户端密钥可以接受检查。这使您能够在应用程序客户端中创建用户、提交密码重置请求和执行其他操作。只有当应用程序是唯一有权访问该密钥的实体时，才必须实现客户端密钥。通常，这在服务器端机密客户端应用程序中是可能的。需要客户端密钥的 M2M 应用程序也是如此。将客户端密钥存储在加密的本地存储器中或 Amazon Secrets Manager。切勿让您的客户机密在公共互联网上可见。

保护其他机密

您使用 Amazon Cognito 用户池的身份验证系统可能会处理私有数据、密码和证书。 Amazon 以下是处理应用程序可能访问的密钥的一些最佳实践。

密码

用户在登录您的应用程序时可能会输入密码。Amazon Cognito 有刷新令牌，您的应用程序可以使用这些令牌在没有新密码提示的情况下继续使用过期的用户会话。不要在本地存储中放置任何密码或密码哈希。将您的应用程序设计为将密码视为不透明，并且仅将其传递到您的用户池。

最佳做法是使用密钥实现无密码身份验证。WebAuthn如果您必须使用密码，请使用安全远程密码 (SRP) 身份验证流程和多因素身份验证 (MFA)。

Amazon 证书

管理身份验证和用户池管理操作需要使用 Amazon 凭据进行身份验证。要在应用程序中实现这些操作，请授予对临时 Amazon 证书的安全访问权限。仅向在您控制的服务器组件上运行的应用程序授予凭据访问权限。不要将包含 Amazon 凭据的应用程序放在公共版本控制系统上，例如。 GitHub 不要在公共客户端应用程序中对 Amazon 凭据进行编码。

PKCE 代码验证器

代码交换（PKCE）的证明密钥用于向您的用户池授权服务器授予 OpenID Connect (OIDC) 授权码。当应用程序请求授权码时，它们会与您的用户池共享代码验证器密钥。要将授权码换成令牌，客户必须重申他们知道代码验证器。这种做法可以防止发行带有被拦截的授权码的代币。

客户端必须为每个授权请求生成一个新的随机码验证器。使用静态或可预测的代码验证器意味着只有这样，攻击者才需要拦截硬编码的验证器和授权码。设计您的应用程序，使其不会向用户公开代码验证器值。

用户池管理最低权限

IAM 策略可以定义委托人对 Amazon Cognito 用户池管理和管理身份验证操作的访问级别。例如：

- 向 Web 服务器授予使用管理 API 操作进行身份验证的权限。
- 对于管理您的 Amazon IAM Identity Center 用户池的用户 Amazon Web Services 账户，请授予用户池维护和报告的权限。

出于IAM策略的目的，Amazon Cognito中的资源粒度级别仅限于两种资源类型：用户池和身份池。请注意，您不能应用权限来管理单个应用程序客户端。在配置用户池时要知道您授予的权限在所有应用程序客户端上均有效。如果您的组织有多个应用程序租户，并且您的安全模型要求租户之间的管理责任分开，则可以实现多租户，每个用户池只有一个租户。

尽管您可以创建具有用户身份验证操作权限的 IAM 策略`InitiateAuth`，但这些权限无效。公开和令牌授权的 API 操作不受 IA M 权限的约束。在可用的用户池身份验证操作中，您只能向服务器端的管理操作授予权限，例如`AdminInitiateAuth`。

您可以使用最低权限列表`Action`来限制用户池的管理级别。以下示例策略适用于可以管理资源服务器 IdPs、应用程序客户端和用户池域的管理员，但不能管理用户或用户池。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserPoolClientAdministrator",
      "Action": [
        "cognito-idp:CreateIdentityProvider",
        "cognito-idp:CreateManagedLoginBranding",
        "cognito-idp:CreateResourceServer",
        "cognito-idp:CreateUserPoolDomain",
        "cognito-idp:DeleteIdentityProvider",
        "cognito-idp:DeleteResourceServer",
        "cognito-idp:DeleteUserPoolDomain",
        "cognito-idp:DescribeIdentityProvider",
        "cognito-idp:DescribeManagedLoginBranding",
        "cognito-idp:DescribeManagedLoginBrandingByClient",
        "cognito-idp:DescribeResourceServer",
        "cognito-idp:DescribeUserPool",
        "cognito-idp:DescribeUserPoolClient",
        "cognito-idp:DescribeUserPoolDomain",
        "cognito-idp:GetIdentityProviderByIdentifier",
```

```
        "cognito-idp:GetUICustomization",
        "cognito-idp:ListIdentityProviders",
        "cognito-idp:ListResourceServers",
        "cognito-idp:ListUserPoolClients",
        "cognito-idp:ListUserPools",
        "cognito-idp:SetUICustomization",
        "cognito-idp:UpdateIdentityProvider",
        "cognito-idp:UpdateManagedLoginBranding",
        "cognito-idp:UpdateResourceServer",
        "cognito-idp:UpdateUserPoolClient",
        "cognito-idp:UpdateUserPoolDomain"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
    }
  ]
}
```

以下示例策略向服务器端应用程序授予用户和组的管理和身份验证。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "UserAdminAuthN",
      "Action": [
        "cognito-idp:AdminAddUserToGroup",
        "cognito-idp:AdminConfirmSignUp",
        "cognito-idp:AdminCreateUser",
        "cognito-idp:AdminDeleteUser",
        "cognito-idp:AdminDeleteUserAttributes",
        "cognito-idp:AdminDisableProviderForUser",
        "cognito-idp:AdminDisableUser",
        "cognito-idp:AdminEnableUser",
        "cognito-idp:AdminForgetDevice",
        "cognito-idp:AdminGetDevice",
        "cognito-idp:AdminGetUser",
        "cognito-idp:AdminInitiateAuth",
        "cognito-idp:AdminLinkProviderForUser",
        "cognito-idp:AdminListDevices",
        "cognito-idp:AdminListGroupsForUser",
        "cognito-idp:AdminListUserAuthEvents",
```

```
        "cognito-idp:AdminRemoveUserFromGroup",
        "cognito-idp:AdminResetUserPassword",
        "cognito-idp:AdminRespondToAuthChallenge",
        "cognito-idp:AdminSetUserMFAPreference",
        "cognito-idp:AdminSetUserPassword",
        "cognito-idp:AdminSetUserSettings",
        "cognito-idp:AdminUpdateAuthEventFeedback",
        "cognito-idp:AdminUpdateDeviceStatus",
        "cognito-idp:AdminUpdateUserAttributes",
        "cognito-idp:AdminUserGlobalSignOut",
        "cognito-idp:AssociateSoftwareToken",
        "cognito-idp:ListGroups",
        "cognito-idp:ListUsers",
        "cognito-idp:ListUsersInGroup",
        "cognito-idp:RevokeToken",
        "cognito-idp:UpdateGroup",
        "cognito-idp:VerifySoftwareToken"
      ],
      "Effect": "Allow",
      "Resource": "arn:aws:cognito-idp:us-west-2:123456789012:userpool/us-
west-2_EXAMPLE"
    }
  ]
}
```

保护和验证令牌

令牌可以包含对群组成员资格的内部引用以及您可能不想向最终用户披露的用户属性。不要将 ID 和访问令牌存储在本地存储中。刷新令牌使用只有您的用户池才能访问的密钥进行加密，并且对用户和应用程序不透明。当用户注@@ 销或出于安全原因您认为不希望保留用户会话时，撤消刷新令牌。

使用访问令牌仅向独立验证令牌有效且未过期的系统授予访问权限。有关验证资源，请参阅验证 JSON 网络令牌。

确定您要信任的身份提供商

使用 SAML 或 OIDC 身份提供商 (IdPs) 配置用户池时， IdPs 您可以创建新用户、设置用户属性和访问您的应用程序资源。SAML 和 OIDC 提供商通常用于 business-to-business (B2B) 或企业场景，在这种场景中，您或您的直属客户控制提供商的成员资格和配置。

社交提供商向互联网上的任何人提供用户帐户，与企业提供商相比，您的控制更少。只有当你准备好允许公众客户登录和访问应用程序 IdPs 中的资源时，才能在应用程序客户端中激活 social。

了解范围对用户个人资料访问权限的影响

您可以在向用户池授权服务器的身份验证请求中请求访问控制范围。这些作用域可以授予您的用户访问外部资源的权限，也可以授予用户查看和修改自己的用户个人资料的权限。配置您的应用程序客户端，使其支持应用程序运行所需的最小范围。

该`aws.cognito.signin.user.admin`作用域存在于 SDK 身份验证通过以下操作颁发的所有访问令牌中InitiateAuth。它专为应用程序中的用户配置文件自助服务操作而设计。您也可以向授权服务器请求此作用域。此作用域是令牌授权的操作（如和）所必需的。UpdateUserAttributesGetUser这些操作的效果受应用程序客户端的读取和写入权限的限制。

`openid`、`profile``email`、和`phone`范围授权向您的用户池授权服务器userInfo 端点上的请求。它们定义端点可以返回的属性。如果请求时没有其他作用域，则作用域会返回所有可用属性，但是当您在请求中请求更多作用域时，响应范围会缩小到由其他作用域表示的属性。`openid``openid`范围还表示对 ID 令牌的请求；当您在向您的请求中省略此范围时，Amazon Cognito 只会发出访问令牌，如果适用，还会发出刷新令牌。对端点授权有关更多信息，请参阅 OpenID Connect 作用域，网址为。应用程序客户端术语

清理用户属性的输入

例如`email`，可能最终成为交付方法和用户名的用户属性有格式限制。其他属性可以有字符串、布尔值或数字数据类型。字符串属性值支持各种输入。配置您的应用程序，以防有人试图向您的用户目录写入不需要的数据或 Amazon Cognito 向用户发送的消息。在将应用程序中用户提交的字符串属性值提交给 Amazon Cognito 之前，对其进行客户端验证。

用户池根据您指定的属性映射将属性从映射 IdPs 到您的用户池。仅将安全且可预测的 IdP 属性映射到用户池字符串属性。

使用 Amazon Cognito 用户池进行身份验证

Amazon Cognito 包含多种对用户进行身份验证的方法。用户可以使用 WebAuthn 密码和密钥登录。Amazon Cognito 可以通过电子邮件或短信向他们发送一次性密码。您可以实现 Lambda 函数来编排您自己的挑战和响应顺序。这些是身份验证流程。在身份验证流程中，用户提供密钥，然后 Amazon Cognito 会验证该密钥，然后发放 JSON 网络令牌 (JWTs) 供应用程序使用 OIDC 库进行处理。在本章中，我们将讨论如何为各种应用程序环境中的各种身份验证流程配置用户池和应用程序客户端。您将了解使用托管登录的托管登录页面的选项，以及在 Amazon SDK 中构建自己的逻辑和前端的选项。

所有用户群体（无论您是否具有域）都可以在用户群体 API 中对用户进行身份验证。如果您向用户群体添加域，则可以使用用户群体端点。用户群体 API 支持针对 API 请求的各种授权模型和请求流程。

为了验证用户的身份，Amazon Cognito 支持除了电子邮件和短信、一次性密码和密钥等密码之外还包含质询类型的身份验证流程。

主题

- 实现身份验证流程
- 关于使用用户池进行身份验证的注意事项
- 身份验证会话示例
- 为托管登录配置身份验证方法
- 在中管理身份验证方法 Amazon SDKs
- 身份验证流程
- API 和 SDK 身份验证的授权模型

实现身份验证流程

无论您是在实现托管登录，还是使用用于身份验证的 S Amazon DK 的自定义应用程序前端，都必须针对要实现的身份验证类型配置应用程序客户端。以下信息描述了应用程序客户端和应用程序中身份验证流程的设置。

App client supported flows

您可以在 Amazon Cognito 控制台中为应用程序客户端配置支持的流程，也可以使用软件开发工具包中的 API 来配置支持的流程。 Amazon 将应用程序客户端配置为支持这些流程后，您可以将其部署到您的应用程序中。

以下过程使用 Amazon Cognito 控制台为应用程序客户端配置可用的身份验证流程。

为身份验证流程配置应用程序客户端（控制台）

1. 登录 Amazon 并导航到 Amazon Cognito 用户池控制台。选择一个用户池或创建一个新的用户池。
2. 在您的用户池配置中，选择应用程序客户端菜单。选择一个应用程序客户端或创建一个新的客户端。
3. 在 “应用程序客户端信息” 下，选择 “编辑”。
4. 在 “应用程序客户端流程” 下，选择要支持的身份验证流程。

为身份验证流程配置应用程序客户端 (API/SDK)

要使用 Amazon Cognito API 为应用程序客户端配置可用的身份验证流程，请在CreateUserPoolClient或UpdateUserPoolClient请求`ExplicitAuthFlows`中设置的值。以下是向客户端提供安全的远程密码 (SRP) 和基于选择的身份验证的示例。

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH",
   "ALLOW_USER_SRP_AUTH
]
```

配置应用程序客户端支持的流程时，可以指定以下选项和 API 值。

应用程序客户端流程支持

身份验证流程	兼容性	控制台	API
基于选择的身份验证	服务器端、客户端	登录时选择身份验证类型	`ALLOW_USER_AUTH`
使用永久密码登录	客户端	使用用户名和密码登录	`ALLOW_USER_PASSWORD_AUTH`
使用永久密码和安全负载登录	服务器端、客户端	使用安全的远程密码 (SRP) 登录	`ALLOW_USER_SRP_AUTH`
刷新令牌	服务器端、客户端	从现有的经过身份验证的会话中获取新的用户令牌	`ALLOW_REFRESH_TOKEN_AUTH`
服务器端身份验证	服务器端	使用服务器端管理凭据登录	`ALLOW_ADMIN_USER_PASSWORD_AUTH`
自定义身份验证	服务器端和客户端定制应用程序。与托管登录不兼容。	使用来自 Lambda 触发器的自定义身份验证流程登录	`ALLOW_CUSTOM_AUTH`

Implement flows in your application

托管登录会自动使您配置的身份验证选项在登录页面中可用。在定制的应用程序中，使用初始流程的声明开始身份验证。

- 要从用户流程选项列表中进行选择，请使用流程声明基于选择的USER_AUTH身份验证。此流程具有在基于客户端的身份验证流程中不可用的可用身份验证方法，例如密钥和无密码身份验证。
- 要预先选择身份验证流程，请使用应用程序客户端中可用的任何其他流程声明基于客户端的身份验证。

当用户登录时，您的InitiateAuth或AdminInitiateAuth请求的正文必须包含一个AuthFlow参数。

基于选择的身份验证：

```
"AuthFlow": "USER_AUTH"
```

使用 SRP 进行基于客户端的身份验证：

```
"AuthFlow": "USER_SRP_AUTH"
```

关于使用用户池进行身份验证的注意事项

在使用 Amazon Cognito 用户池设计身份验证模型时，请考虑以下信息。

托管登录和托管 UI 中的身份验证流程

托管登录和经典托管用户界面有不同的身份验证选项。在托管登录中，您只能进行无密码和密钥身份验证。

自定义身份验证流程仅在 Amazon SDK 身份验证中可用

您无法使用托管登录或经典托管 UI 进行自定义身份验证流程，也无法使用 Lambda 触发器进行自定义身份验证。自定义身份验证可在使用进行身份验证时使用 Amazon SDKs。

外部身份提供商 (IdP) 登录的托管登录

在使用身份验证时，您无法通过第三方 IdPs登录用户 Amazon SDKs。您必须实现托管登录或经典托管用户界面，重定向到 IdPs，然后在应用程序中使用 OIDC 库处理生成的身份验证对象。有关托管登录的更多信息，请参阅用户池托管登录。

无密码身份验证对其他用户功能的影响

使用用户池和应用程序客户端中的一次性密码或密钥激活无密码登录会影响用户的创建和迁移。启用无密码登录时：

1. 管理员可以创建没有密码的用户。默认的邀请消息模板更改为不再包含`{###}`密码占位符。有关更多信息，请参阅 以管理员身份创建用户账户。
2. 对于基于 SDK 的SignUp操作，用户在注册时无需提供密码。即使允许使用无密码身份验证，托管登录和托管用户界面也需要在注册页面中输入密码。有关更多信息，请参阅 注册并确认用户账户。
3. 从 CSV 文件导入的用户可以立即使用无密码选项登录，而无需重置密码，前提是他们的属性包括可用无密码登录选项的电子邮件地址或电话号码。有关更多信息，请参阅 通过 CSV 文件将用户导入用户池中。
4. 无密码身份验证不会调用用户迁移 Lambda 触发器。
5. 使用无密码第一因素登录的用户无法在会话中添加多重身份验证 (MFA) 因素。只有基于密码的身份验证流程支持 MFA。

Passkey 依赖方 URLs 不能出现在公共后缀列表中

在密钥配置中，您可以使用自己拥有的域名作为信赖方 (RP) ID。`www.example.com`此配置旨在支持在您拥有的域上运行的自定义应用程序。公共后缀列表 (PSL) 包含受保护的高级域。当你尝试将 RP 网址设置为 PSL 上的域名时，Amazon Cognito 会返回错误。

主题

- 身份验证会话流持续时间
- 登录尝试失败时的锁定行为

身份验证会话流持续时间

根据用户池的功能，您最终可能会在应用程序从 Amazon Cognito 检索令牌`RespondToAuthChallenge`之前或之前对`InitiateAuth`多个挑战做出响应。Amazon Cognito 在对每个请求的响应中都包含一个会话字符串。要将您的 API 请求合并到身份验证流程中，请在每个后续请求中包含来自上一个请求的响应中的会话字符串。默认情况下，在会话字符串过期之前，您的用户有三分钟时间完成每项质询。要调整此时段，请更改您的应用程序客户端的 Authentication flow session duration（身份验证流程会话持续时间）。以下过程介绍如何在应用程序客户端配置中更改此设置。

> **Note**
>
> 身份验证流程会话持续时间设置适用于使用 Amazon Cognito 用户群体 API 进行身份验证。托管登录将多因素身份验证的会话持续时间设置为 3 分钟，密码重置代码的会话持续时间设置为 8 分钟。

Amazon Cognito console

配置应用程序客户端身份验证流程会话持续时间 (Amazon Web Services Management Console)

1. 在您的用户群体中的 App integration（应用程序集成）选项卡上，从 App clients and analytics（应用程序客户端和分析）容器中选择您的应用程序客户端的名称。
2. 在应用程序客户端信息容器中选择编辑。
3. 将身份验证流程会话持续时间的值更改为您想要的 SMS 和电子邮件 MFA 代码的有效时长，以分钟为单位。这还会更改任何用户在您的应用程序客户端中必须完成任何身份验证质询的时间。
4. 选择保存更改。

User pools API

配置应用程序客户端身份验证流程会话持续时间 (Amazon Cognito API)

1. 使用 `DescribeUserPoolClient` 请求中您的现有用户群体设置准备 `UpdateUserPoolClient` 请求。您的 `UpdateUserPoolClient` 请求必须包含所有现有的应用程序客户端属性。
2. 将 `AuthSessionValidity` 的值更改为 SMS MFA 代码所需的有效期，以分钟为单位。这还会更改任何用户在您的应用程序客户端中必须完成任何身份验证质询的时间。

有关应用程序客户端的更多信息，请参阅特定于应用程序的应用程序客户端设置。

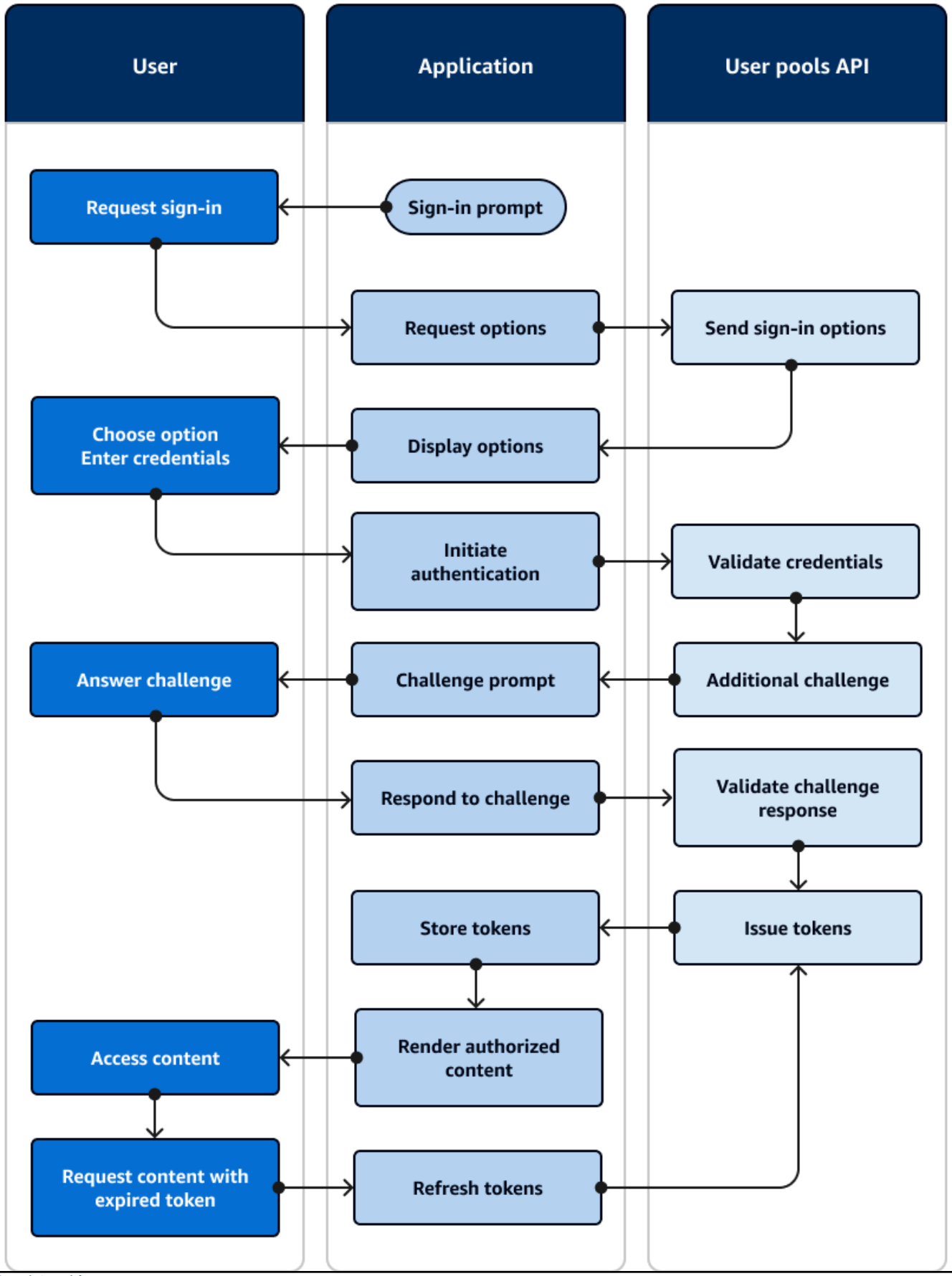
登录尝试失败时的锁定行为

在使用密码进行五次未经身份验证或 IAM 授权的登录尝试失败后，Amazon Cognito 会将您的用户锁定一秒钟。然后，每多一次失败的尝试，锁定持续时间将增加一倍，最长约为 15 分钟。在锁定期内进行的尝试会产生 `Password attempts exceeded` 异常，不会影响后续锁定期的持续时间。对于累计 n 次的失败登录尝试（不包括 `Password attempts exceeded` 异常），Amazon Cognito 会将您的用

户锁定 2^(n-5) 秒。要将锁定重置为其 n=0 初始状态，用户必须在锁定期到期后才能成功登录，或者在锁定后连续 15 分钟的任何时间内都不得发起任何登录尝试。此行为随时可能会发生变化。此行为不适用于自定义质询，除非它们还执行基于密码的身份验证。

身份验证会话示例

下图和 step-by-step指南说明了用户登录应用程序的典型场景。示例应用程序为用户提供了多个登录选项。他们通过输入凭据来选择一个，提供额外的身份验证因素，然后登录。


User
Application
User pools API
Request sign-in
Sign-in prompt
Request options
Send sign-in options
Choose option
Enter credentials
Display options
Initiate authentication
Validate credentials
Answer challenge
Challenge prompt
Additional challenge
Respond to challenge
Validate challenge response
Store tokens
Issue tokens
Access content
Render authorized content
Request content with expired token
Refresh tokens

想象一下带有登录页面的应用程序，用户可以在其中使用用户名和密码登录、在电子邮件中请求一次性验证码或选择指纹选项。

1. 登录提示：您的应用程序显示带有 “登录” 按钮的主屏幕。
2. 请求登录：用户选择 “登录”。您的应用程序会从 Cookie 或缓存中检索他们的用户名，或者提示他们输入用户名。
3. 请求选项：您的应用通过`USER_AUTH`流程中的 `InitiateAuth` API 请求来请求用户的登录选项，请求用户可用的登录方法。
4. 发送登录选项：Amazon Cognito 回复`PASSWORD`为`EMAIL_OTP`、和。`WEB_AUTHN`该响应包含一个会话标识符，供您在下一个响应中回放。
5. 显示选项：您的应用程序显示用户界面元素，供用户输入用户名和密码、获取一次性代码或扫描指纹。
6. 选择选项/输入凭证：用户输入其用户名和密码。
7. 启动身份验证：您的应用程序通过一个 `RespondToAuthChallenge` API 请求向用户的登录信息提供确认用户名密码登录并提供用户名和密码。
8. 验证凭证：Amazon Cognito 确认用户的证书。
9. 其他挑战：用户使用身份验证器应用程序配置了多因素身份验证。亚马逊 Cognito 回来了一项挑战。`SOFTWARE_TOKEN_MFA`
10. 质询提示：您的应用程序会显示一个表单，要求用户的身份验证器应用程序提供基于时间的一次性密码 (TOTP)。
11. 回答挑战：用户提交 TOTP。
12. 回应质疑：在另一个`RespondToAuthChallenge`请求中，您的应用程序会提供用户的 TOTP。
13. 验证质询响应：Amazon Cognito 确认用户的代码，并确定您的用户池已配置为不向当前用户发出其他质询。
14. 发行令牌：Amazon Cognito 会返回 ID、访问权限和刷新 JSON 网络令牌 () JWTs。用户的初始身份验证已完成。
15. 存储令牌：您的应用程序会缓存用户的令牌，以便它可以引用用户数据，授权访问资源，并在令牌到期时更新令牌。
16. 呈现授权内容：您的应用程序根据用户的身份和角色确定其对资源的访问权限，并提供应用程序内容。
17. 访问内容：用户已登录并开始使用该应用程序。
18. 使用过期令牌请求内容：稍后，用户请求需要授权的资源。用户的缓存令牌已过期。

19刷新令牌：您的应用程序使用用户保存的刷新令牌`InitiateAuth`发出请求。

20发行令牌：亚马逊 Cognito 会返回新的 ID 和访问权限。 JWTs用户会话可以安全刷新，无需额外提示输入凭据。

您可以使用Amazon Lambda 触发器来自定义用户身份验证的方式。作为身份验证流程的一部分，这些触发器将发布并验证自己的质询。

您还可以在安全后端服务器上对用户使用管理员身份验证流程。您可以使用用户迁移身份验证流程来实现用户迁移，而无需用户重置密码。

为托管登录配置身份验证方法

当您希望用户登录、注销或重置密码时，您可以调用托管登录页面（用于用户池身份验证的 Web 前端）。在此模型中，您的应用程序导入 OIDC 库，以使用用户池托管登录页面处理基于浏览器的身份验证尝试。可供用户使用的身份验证形式取决于您的用户池和应用程序客户端的配置。在您的应用程序客户端中实现该`ALLOW_USER_AUTH`流程，Amazon Cognito 会提示用户从可用选项中选择登录方法。实施`ALLOW_USER_PASSWORD_AUTH`并分配 SAML 提供商，您的登录页面会提示用户选择输入用户名和密码或与其 IdP 连接。

Amazon Cognito 用户池控制台可以帮助您开始为应用程序设置托管登录身份验证。创建新的用户池时，请指定您要开发的平台，控制台将为您提供实现 OIDC 的示例，以及带有用于实现登录和注销流程的入门代码的 OAuth 库。您可以使用许多 OIDC 依赖方实现来构建托管登录。我们建议您尽可能使用经过认证的 OIDC 信赖方库。有关更多信息，请参阅 用户池入门。

通常，OIDC 信赖方库会定期检查用户池的`.well-known/openid-configuration`端点以确定发行者， URLs 例如令牌端点和授权端点。作为最佳实践，在必须选择的情况下实现这种自动发现行为。手动配置发行者端点会引发错误。例如，您可以更改您的用户池域。路径`openid-configuration`未链接到您的用户池域，因此自动发现服务端点的应用程序将自动获取您的域名更改。

托管登录的用户池设置

您可能希望允许通过多个提供商登录您的应用程序，或者您可能希望将 Amazon Cognito 用作独立的用户目录。您可能还想收集用户属性、设置和提示使用 MFA，或者要求将电子邮件地址作为用户名。您无法直接编辑托管登录和托管用户界面中的字段。相反，用户池的配置会自动设置托管登录身份验证流程的处理。

以下用户池配置项目决定了 Amazon Cognito 在托管登录和托管用户界面中向用户提供的身份验证方法。

User pool options (Sign-in menu)

以下选项位于 Amazon Cognito 控制台用户池的登录菜单中。

Cognito 用户池登录选项

有用户名选项。您的托管登录和托管用户界面页面仅接受您选择的格式的用户名。例如，当您设置以电子邮件作为唯一登录选项的用户池时，您的托管登录页面仅接受电子邮件格式的用户名。

必需的属性

当您在用户池中根据需要设置属性时，托管登录会在用户注册时提示他们输入该属性的值。

基于选择的登录选项

中包含身份验证方法的设置基于选择的身份验证。在这里，您可以打开或关闭诸如密钥和无密码之类的身份验证方法。这些方法仅适用于拥有托管登录域和功能计划高于 Lite 级别的用户池。

多重身份验证

托管登录和托管用户界面处理 MFA 的注册和身份验证操作。当您的用户池中需要使用 MFA 时，您的登录页面会自动提示用户设置其附加系数。它们还会提示拥有 MFA 配置的用户使用 MFA 代码完成身份验证。当 MFA 处于关闭状态或在用户池中处于可选状态时，您的登录页面不会提示您设置 MFA。

恢复用户账户

用户池的自助服务账户恢复设置决定了您的登录页面是否显示用户可以重置密码的链接。

User pool options (Domain menu)

以下选项位于 Amazon Cognito 控制台用户池的域名菜单中。

域

您选择的用户池域将设置用户在调用浏览器进行身份验证时打开的链接路径。

品牌版本

您对品牌版本的选择决定了您的用户池域是显示托管登录还是托管用户界面。

User pool options (Social and external providers menu)

以下选项位于 Amazon Cognito 控制台用户池的社交和外部提供商菜单中。

提供商

对于用户池中的每个应用程序客户端，您添加到用户池中的身份提供者 (IdPs) 可以处于活动状态或非活动状态。

App client options

以下选项位于 Amazon Cognito 控制台用户池的应用程序客户端菜单中。要查看这些选项，请从列表中选择一个应用程序客户端。

快速设置指南

快速设置指南包含适用于各种开发人员环境的代码示例。它们包含将托管登录身份验证与您的应用程序集成所需的库。

应用程序客户端信息

编辑此配置以设置 IdPs 为当前应用程序客户端所代表的应用程序分配的设置。在托管登录页面上，Amazon Cognito 会为用户显示选择。这些选择由分配的方法和 IdP 决定。例如，如果您分配了一个名为的 SAML 2.0 IdP 和本地用户池登录`MySAML`名，则您的托管登录页面会显示身份验证方法提示和一个按钮。`MySAML`

身份验证设置

编辑此配置以设置应用程序的身份验证方法。在托管登录页面上，Amazon Cognito 会为用户显示选择。这些选择取决于用户池作为 IdP 的可用性以及您分配的方法。例如，如果您分配了基于选择的`ALLOW_USER_AUTH`身份验证，则您的托管登录页面会显示可用选项，例如输入电子邮件地址和使用密钥登录。托管登录页面还会为分配的用户呈现按钮 IdPs。

登录页面

使用此选项卡中的可用选项，设置托管登录或托管用户界面用户交互页面的视觉效果。有关更多信息，请参阅 将品牌应用于托管登录页面。

在中管理身份验证方法 Amazon SDKs

Amazon Cognito 用户池中的用户可以使用各种初始登录选项或因素进行登录。对于某些因素，用户可以跟进多因素身份验证 (MFA)。这些首要因素包括用户名和密码、一次性密码、密钥和自定义身份验证。有关更多信息，请参阅 身份验证流程。当您的应用程序具有内置 UI 组件并导入 S Amazon DK 模块时，您必须构建用于身份验证的应用程序逻辑。您必须选择两种主要方法之一，然后从该方法中选择要实现的身份验证机制。

您可以实现基于客户端的身份验证，让您的应用程序或客户端预先声明身份验证类型。您的另一种选择是基于选择的身份验证，即您的应用程序收集用户名并请求用户可用的身份验证类型。你可以在同一个

应用程序中一起实现这些模型，也可以根据自己的要求在应用程序客户端之间拆分。每种方法都有其独有的功能，例如基于客户端的自定义身份验证和基于选择的无密码身份验证。

在使用用户池 API 的 Amazon SDK 实现进行身份验证的自定义应用程序中，您的 API 请求结构必须与用户池配置、应用程序客户端配置和客户端首选项保持一致。以 o `InitiateAuth` f 开头的会话`USER_AUTH`开始基于选择`AuthFlow`的身份验证。Amazon Cognito 以首选身份验证方法或选项列表来回应您的 API。以 of 开头`AuthFlow`的会话直接`CUSTOM_AUTH`进入使用 Lambda 触发器的自定义身份验证。

有些身份验证方法固定为两种流程类型中的一种，有些方法在两种流量类型中都可用。

主题

- 基于选择的身份验证
- 基于客户端的身份验证

基于选择的身份验证

您的应用程序可以在基于选择的身份验证中请求以下身份验证方法。在InitiateAuth或的`PREFERRED_CHALLENGE`参数中声明这些选项 AdminInitiateAuth，或者在RespondToAuthChallenge或的`ChallengeName`参数中声明这些选项AdminRespondToAuthChallenge。

1. `EMAIL_OTP` 和 `SMS_OTP`

   使用一次性密码进行无密码登录

2. `WEB_AUTHN`

   使用密钥进行无密码登录 WebAuthn

3. `PASSWORD`

   使用永久密码登录

   使用永久密码和安全负载登录

   登录后的 MFA

要在其 API 上下文中查看这些选项，请参阅`ChallengeName`中的RespondToAuthChallenge。

基于选择的登录会根据您的初始请求发出质疑。此质询要么验证所请求的选项是否可用，要么提供可用选项列表。您的应用程序可以向用户显示这些选择，然后用户输入其首选登录方法的凭据，然后在质询响应中继续进行身份验证。

您的身份验证流程中有以下基于选择的选项。所有此类请求都要求您的应用程序首先收集用户名或从缓存中检索用户名。

1. USERNAME仅使用 AuthParameters of 请求选项。亚马逊 Cognito 回来了一项挑战。SELECT_CHALLENGE然后，您的应用程序可以提示用户选择质询并将此响应返回到您的用户池。
2. 使用 of PREFERRED_CHALLENGE 和首选挑战AuthParameters的参数（如果有）请求首选质询。例如，如果您请求 a PREFERRED_CHALLENGEPASSWORD_SRP，则还必须包括SRP_A。如果您的用户、用户池和应用程序客户端都针对首选挑战进行了配置，则 Amazon Cognito 会以该挑战的下一步作为响应，例如PASSWORD_VERIFIER在PASSWORD_SRP流程中或EMAIL_OTP和SMS_OTP流程CodeDeliveryDetails中。如果首选挑战不可用，Amazon Cognito 会回复SELECT_CHALLENGE并列出可用挑战列表。
3. 首先让用户登录，然后请求他们选择的身份验证选项。使用已登录用户的访问令牌的GetUserAuthFactors请求会返回他们可用的基于选择的身份验证因素和 MFA 设置。使用此选项，用户可以先使用用户名和密码登录，然后激活其他形式的身份验证。您还可以使用此操作为使用首选挑战登录的用户查看其他选项。

要将您的应用程序客户端配置为基于选择的身份验证，请ALLOW_USER_AUTH添加到允许的身份验证流程中。您还必须选择要在用户池配置中允许的基于选择的因素。以下过程说明如何选择基于选择的身份验证因素。

Amazon Cognito console

在用户池中配置基于选择的身份验证选项

1. 登录 Amazon 并导航到 Amazon Cognito 用户池控制台。选择一个用户池或创建一个新的用户池。
2. 在您的用户池配置中，选择登录菜单。找到 “基于选择的登录选项”，然后选择 “编辑”。
3. 密码选项始终可用。这包括PASSWORD和PASSWORD_SRP流程。选择要添加到用户选项中的其他选项。您可以为添加 PasskeyWEB_AUTHN，为添加电子邮件一次性密码EMAIL_OTP，为添加短信一次性密码。SMS_OTP
4. 选择保存更改。

API/SDK

以下部分CreateUserPool或UpdateUserPool请求正文配置了基于选择的身份验证的所有可用选项。

```
"Policies": {
    "SignInPolicy": {
        "AllowedFirstAuthFactors": [
            "PASSWORD",
            "WEB_AUTHN",
            "EMAIL_OTP",
            "SMS_OTP"
        ]
    }
},
```

基于客户端的身份验证

基于客户端的身份验证支持以下身份验证流程。在InitiateAuth或的AuthFlow参数中声明这些选项AdminInitiateAuth。

1. USER_PASSWORD_AUTH 和 ADMIN_USER_PASSWORD_AUTH

   使用永久密码登录

   登录后的 MFA

   此身份验证流程等同PASSWORD于基于选择的身份验证。

2. USER_SRP_AUTH

   使用永久密码和安全负载登录

   登录后的 MFA

   此身份验证流程等同PASSWORD_SRP于基于选择的身份验证。

3. REFRESH_TOKEN_AUTH

   刷新令牌

   此身份验证流程仅在基于客户端的身份验证中可用。

4. CUSTOM_AUTH

自定义身份验证

此身份验证流程仅在基于客户端的身份验证中可用。

对于基于客户端的身份验证，Amazon Cognito 假设您已经确定了用户在开始身份验证流程之前想要如何进行身份验证。确定用户想要提供的登录系数的逻辑必须使用默认设置或自定义提示来确定，然后在用户池的第一个请求中声明。例如USER_SRP_AUTH，该InitiateAuth请求声明AuthFlow的登录与列出的选项之一直接对应。通过此声明，请求还包括开始身份验证的参数，例如USERNAMESECRET_HASH、和SRP_A。在此请求之后，Amazon Cognito 可能会提出其他挑战，PASSWORD_VERIFIER例如 SRP 或使用 TOTP MFA SOFTWARE_TOKEN_MFA 进行密码登录。

要将您的应用程序客户端配置为基于客户端的身份验证，请添加除允许的身份验证流程之外ALLOW_USER_AUTH的所有身份验证流程。例如ALLOW_USER_PASSWORD_AUTH、ALLOW_CUSTOM_AUTH、ALLOW_REFRESH_TOKEN_AUTH。要允许基于客户端的身份验证流程，无需进行额外的用户池配置。

身份验证流程

使用 Amazon Cognito 用户池进行身份验证的过程最好描述为用户做出初步选择、提交凭证和应对其他挑战的流程。当您在应用程序中实施托管登录身份验证时，Amazon Cognito 会管理这些提示和挑战的流程。当您在应用程序后端使用 S Amazon DK 实现流程时，必须构造请求逻辑，提示用户输入并响应挑战。

作为应用程序管理员，您的用户特征、安全要求和授权模型有助于确定您希望如何允许用户登录。问问自己以下问题。

- 我是否要允许用户使用其他身份提供者的凭据登录 (IdPs)？
- 用户名和密码是否足以证明身份？
- 我的用户名密码身份验证请求能否被拦截？ 我希望我的应用程序传输密码，还是使用哈希和盐来协商身份验证？
- 我是否要允许用户跳过密码输入并接收一次性密码来登录？
- 我是否允许用户使用指纹、面孔或硬件安全密钥登录？
- 如果有的话，我何时需要多因素身份验证 (MFA)？
- 我是否要在不重新提示输入凭据的情况下保留用户会话？
- 我是否想将我的授权模式扩展到 Amazon Cognito 的内置功能之外？

获得这些问题的答案后，您可以学习如何激活相关功能并在应用程序发出的身份验证请求中实现这些功能。

为用户设置登录流程后，您可以通过 API 操作请求来检查其当前状态的 MFA 和基于选择的身份验证因素。GetUserAuthFactors此操作需要使用登录用户的访问令牌进行授权。它返回用户身份验证因素和 MFA 设置。

主题

- 使用第三方登录 IdPs
- 使用永久密码登录
- 使用永久密码和安全负载登录
- 使用一次性密码进行无密码登录
- 使用密钥进行无密码登录 WebAuthn
- 登录后的 MFA
- 刷新令牌
- 自定义身份验证
- 用户迁移身份验证流程

使用第三方登录 IdPs

Amazon Cognito 用户池充当 “用苹果登录”、“用亚马逊登录” 和 OpenID Connect (OIDC) 服务之间的 IdPs身份验证会话的中间代理。此过程也称为联合登录或联合身份验证。联合身份验证不使用您可以在应用程序客户端中内置的任何身份验证流程。相反，您可以将已配置的用户池分配 IdPs 给您的应用程序客户端。当用户在托管登录中选择自己的 IdP 或者您的应用程序调用会话并重定向到其 IdP 登录页面时，就会发生联合登录。

通过联合登录，您可以将主身份验证和 MFA 身份验证因素委托给用户的 IdP。Amazon Cognito 不会将本节中的其他高级流程添加到联合用户，除非您将这些流程关联到本地用户。未关联的联合用户有用户名，但它们是映射的属性数据的存储，通常不用于独立于基于浏览器的流程的登录。

实施资源

- 用户池使用第三方身份提供商登录

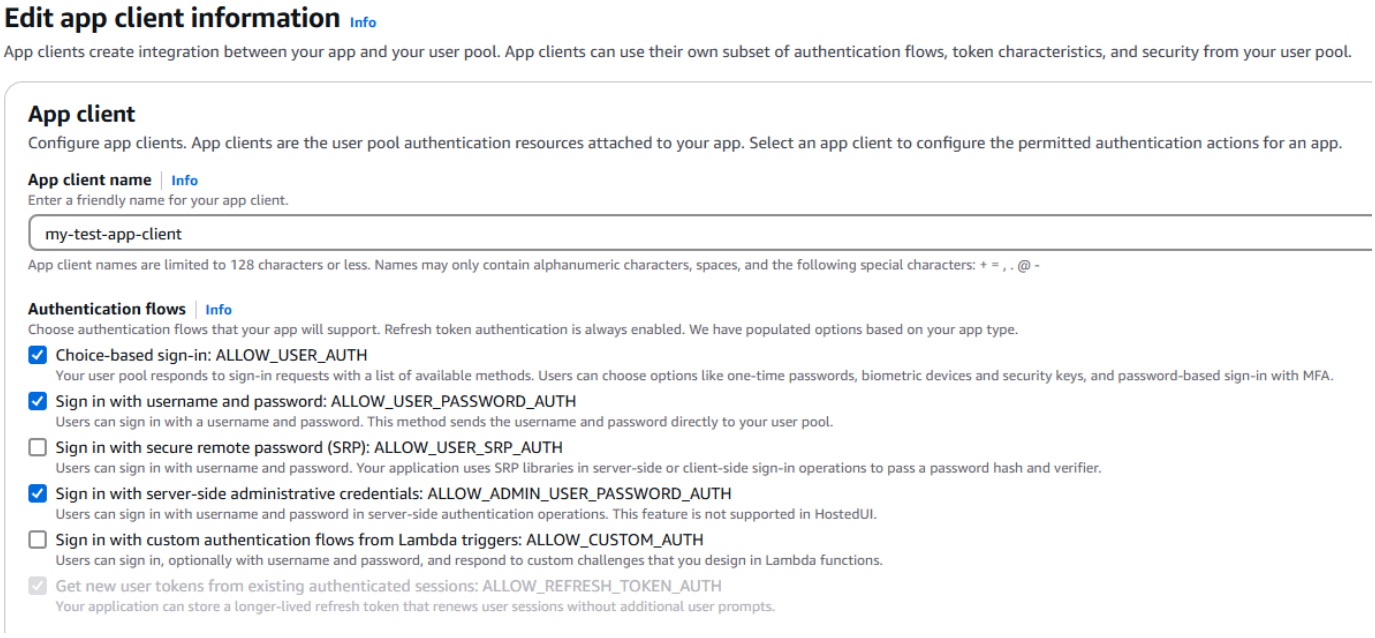
使用永久密码登录

在 Amazon Cognito 用户池中，每个用户都有一个用户名。这可能是电话号码、电子邮件地址或选定的或管理员提供的标识符。此类用户可以使用其用户名和密码登录，也可以选择提供 MFA。用户池可以使用公共或 IAM 授权的 API 操作和 SDK 方法执行用户名密码登录。您的应用程序可以直接将密码发送到您的用户池进行身份验证。您的用户池会使用其他质询或 JSON Web 令牌 (JWTs) 作为成功身份验证的结果进行响应。

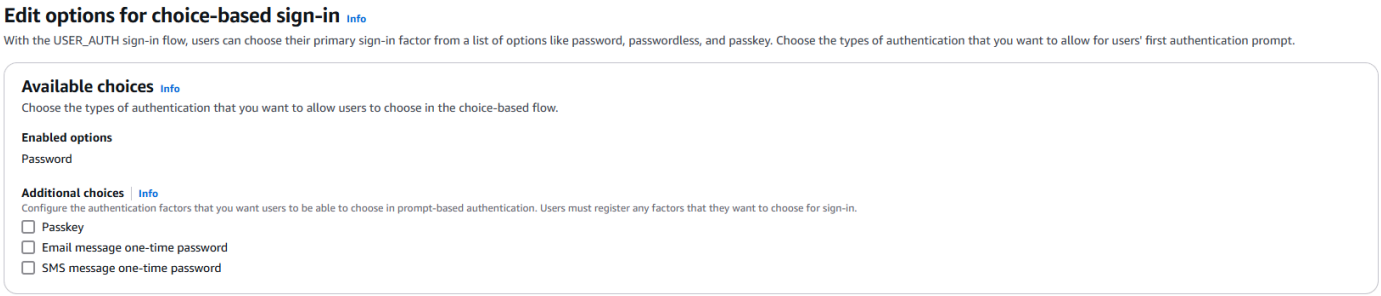
Activate password sign-in

要使用用户名和密码激活基于客户端的身份验证，请将您的应用程序客户端配置为允许。在 Amazon Cognito 控制台中，导航到用户池配置中 “应用程序” 下的 “应用程序客户端” 菜单。要允许客户端移动或本机应用程序使用普通密码登录，请编辑应用程序客户端，然后在 “身份验证流程” 下选择 “使用用户名和密码登录：ALLOW_USER_PAS SWORD_AUTH”。要允许服务器端应用程序使用普通密码登录，请编辑应用程序客户端，然后选择使用服务器端管理凭据登录：ALLOW_ADMIN_USER_PASSWORD_AUTH。

要使用用户名和密码激活基于选择的身份验证，请将您的应用程序客户端配置为允许。编辑您的应用程序客户端，然后选择基于选择的登录：ALLOW_USER_ AUTH。



要验证密码身份验证在基于选择的身份验证流程中是否可用，请导航至 “登录” 菜单并查看 “基于选择的登录选项” 下的部分。如果密码在 “可用选项” 下可见，则可以使用纯密码身份验证登录。“密码” 选项包括普通和 SRP 用户名密码身份验证变体。



在CreateUserPoolClient或UpdateUserPoolClient请求中`ExplicitAuthFlows`使用您的首选username-and-password身份验证选项进行配置。

```
"ExplicitAuthFlows": [
   "ALLOW_USER_PASSWORD_AUTH",
   "ALLOW_ADMIN_USER_PASSWORD_AUTH",
   "ALLOW_USER_AUTH"
]
```

在CreateUserPool或UpdateUserPool请求中，`Policies`使用要支持的基于选择的身份验证流程进行配置。中的`PASSWORD`值`AllowedFirstAuthFactors`包括纯密码和 SRP 身份验证流程选项。

```
"Policies": {
   "SignInPolicy": {
      "AllowedFirstAuthFactors": [
         "PASSWORD",
         "EMAIL_OTP",
         "WEB_AUTHN"
      ]
   }
}
```

Choice-based sign-in with a password

要使用用户名密码身份验证让用户登录应用程序，请按以下方式配置您的AdminInitiateAuth或InitiateAuth请求的正文。如果当前用户有资格进行用户名密码身份验证，则此登录请求成功或继续下一个质询。否则，它会以可用的主要因素身份验证质询列表作为响应。这组参数是登录所需的最低参数。还有其他参数可用。

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
```

```
      "PREFERRED_CHALLENGE" : "PASSWORD",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
}
```

您也可以省略该PREFERRED_CHALLENGE值并收到包含用户合格登录因素列表的响应。

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser"
   },
   "ClientId": "1example23456789"
}
```

如果您没有提交首选挑战或提交的用户没有资格参加首选挑战，Amazon Cognito 会返回选项列表。AvailableChallenges如果AvailableChallenges包含 o ChallengeName PASSWORD f，则可以使用以下格式的RespondToAuthChallenge或AdminRespondToAuthChallenge质询响应继续进行身份验证。您必须传递一个Session参数，该参数将质询响应与 API 响应关联到您的初始登录请求。这组参数是登录所需的最低参数。还有其他参数可用。

```
{
   "ChallengeName": "PASSWORD",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's Password]"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response"
}
```

Amazon Cognito 使用令牌或其他必需的质PASSWORD询（例如多因素身份验证 (MFA)）来响应符合条件且成功的首选质询请求和质询响应。

Client-based sign-in with a password

要使用用户名密码身份验证让用户登录到客户端应用程序，请按以下方式配置请求的正文。InitiateAuth这组参数是登录所需的最低参数。还有其他参数可用。

```
{
```

```
   "AuthFlow": "USER_PASSWORD_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
}
```

要使用用户名密码身份验证让用户登录服务器端应用程序，请按以下方式配置请求的正文。AdminInitiateAuth您的应用程序必须使用 Amazon 凭证签署此请求。这组参数是登录所需的最低参数。还有其他参数可用。

```
{
   "AuthFlow": "ADMIN_USER_PASSWORD_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PASSWORD" : "[User's password]"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito 使用令牌或其他必需的质询（例如多因素身份验证 (MFA)）来响应成功的请求。

使用永久密码和安全负载登录

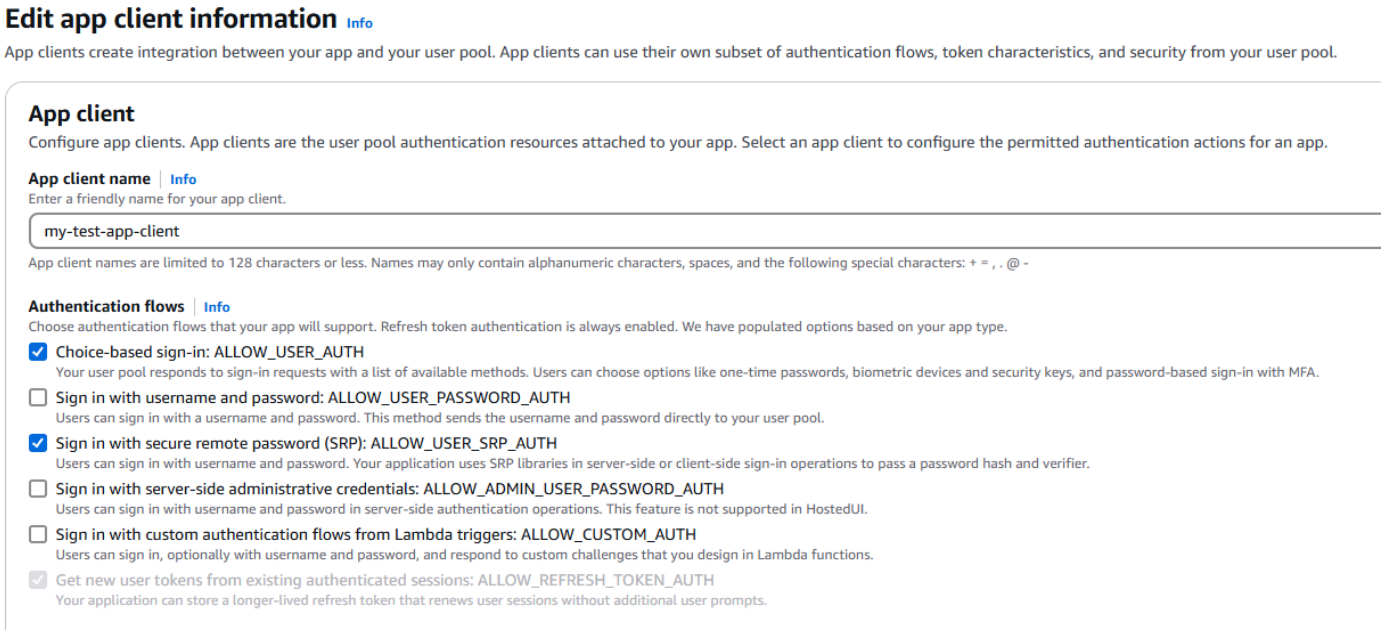
用户池中用户名密码登录方法的另一种形式是使用安全远程密码 (SRP) 协议。此选项发送知道密码的证据（密码哈希和盐），您的用户池可以验证该密码。由于向 Amazon Cognito 发出的请求中没有可读的机密信息，因此您的应用程序是处理用户输入密码的唯一实体。SRP 身份验证涉及数学计算，最好由可以导入 SDK 中的现有组件来完成。SRP 通常在客户端应用程序（如移动应用程序）中实现。有关该协议的更多信息，请参阅斯坦福大学SRP主页。维基百科也有资源和示例。有各种公共库可用于为您的身份验证流程执行 SRP 计算。

Amazon Cognito 身份验证 initiate-challenge-respond顺序使用 SRP 验证用户及其密码。您必须将用户池和应用程序客户端配置为支持 SRP 身份验证，然后在应用程序中实现登录请求和质询响应的逻辑。您的 SRP 库可以生成随机数和计算值，向您的用户池证明您拥有用户的密码。您的应用程序将这些计算值填入 JSON 格式`AuthParameters`的 Amazon Cognito 用户池 API 操作和 SDK 身份验证方法中的`ChallengeParameters`字段。

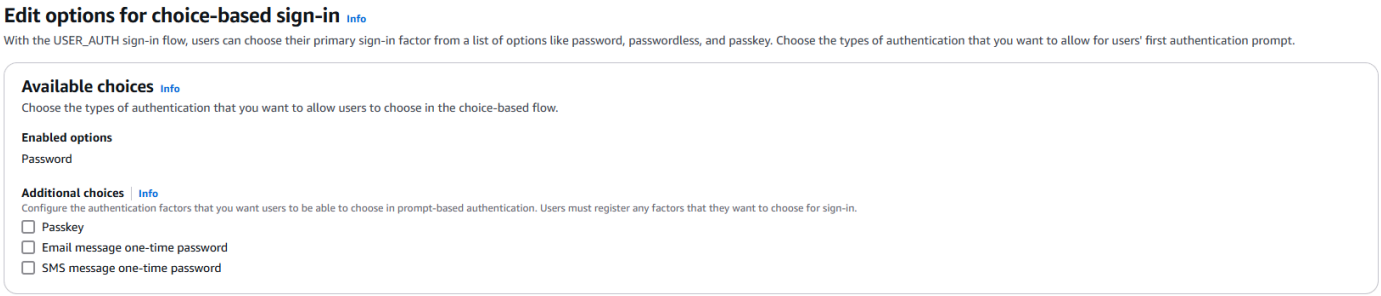
Activate SRP sign-in

要使用用户名和 SRP 激活基于客户端的身份验证，请将应用程序客户端配置为允许。在 Amazon Cognito 控制台中，导航到用户池配置中 “应用程序” 下的 “应用程序客户端” 菜单。要允许客户端移动或本机应用程序使用 SRP 登录，请编辑应用程序客户端，然后在 “身份验证流程” 下选择 “使用安全远程密码 (SRP) 登录：ALLOW_USER_SRP_ AUTH”。

要使用用户名和 SRP 激活基于选择的身份验证，请编辑您的应用程序客户端，然后选择基于选择的登录：ALLOW_ USER_AU TH。



要验证基于选择的身份验证流程中是否可用 SRP 身份验证，请导航至 “登录” 菜单并查看 “基于选择的登录选项” 下的部分。如果密码在 “可用选项” 下可见，则可以使用 SRP 身份验证登录。密码选项包括纯文本和 SRP 用户名密码身份验证变体。



在CreateUserPoolClient或UpdateUserPoolClient请求中`ExplicitAuthFlows`使用您的首选 username-and-password身份验证选项进行配置。

```
"ExplicitAuthFlows": [
```

```
   "ALLOW_USER_SRP_AUTH",
   "ALLOW_USER_AUTH"
]
```

在CreateUserPool或UpdateUserPool请求中，Policies使用要支持的基于选择的身份验证流程进行配置。中的PASSWORD值AllowedFirstAuthFactors包括纯文本密码和 SRP 身份验证流程选项。

```
"Policies": {
   "SignInPolicy": {
      "AllowedFirstAuthFactors": [
         "PASSWORD",
         "EMAIL_OTP",
         "WEB_AUTHN"
      ]
   }
}
```

Choice-based sign-in with SRP

要让用户通过 SRP 使用用户名密码身份验证登录应用程序，请按以下方式配置您的AdminInitiateAuth或InitiateAuth请求的正文。如果当前用户有资格进行用户名密码身份验证，则此登录请求成功或继续下一个质询。否则，它会以可用的主要因素身份验证质询列表作为响应。这组参数是登录所需的最低参数。还有其他参数可用。

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PREFERRED_CHALLENGE" : "PASSWORD_SRP",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789"
}
```

您也可以省略该PREFERRED_CHALLENGE值并收到包含用户合格登录因素列表的响应。

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser"
```

```
   },
   "ClientId": "1example23456789"
}
```

如果您没有提交首选挑战或提交的用户没有资格参加首选挑战，Amazon Cognito 会返回选项列表。`AvailableChallenges`如果`AvailableChallenges`包含 o `ChallengeName` `PASSWORD_SRP` f，则可以使用以下格式的RespondToAuthChallenge或AdminRespondToAuthChallenge质询响应继续进行身份验证。您必须传递一个`Session`参数，该参数将质询响应与 API 响应关联到您的初始登录请求。这组参数是登录所需的最低参数。还有其他参数可用。

```
{
   "ChallengeName": "PASSWORD_SRP",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response"
}
```

Amazon Cognito 通过质询来回应符合条件的首选质询请求和`PASSWORD_SRP`质疑回复。`PASSWORD_VERIFIER`您的客户必须完成 SRP 计算并回应RespondToAuthChallenge或AdminRespondToAuthChallenge请求中的质疑。

```
{
   "ChallengeName": "PASSWORD_VERIFIER",
   "ChallengeResponses": {
      "PASSWORD_CLAIM_SIGNATURE" : "string",
      "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
      "TIMESTAMP" : "string"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

成功响应`PASSWORD_VERIFIER`质询后，Amazon Cognito 会发放令牌或其他必需的质询，例如多因素身份验证 (MFA)。

Client-based sign-in with SRP

SRP 身份验证在客户端身份验证中比在服务器端更常见。但是，您可以将 SRP 身份验证与InitiateAuth和AdminInitiateAuth一起使用。要让用户登录应用程序，请按如下方式配置您的InitiateAuth或AdminInitiateAuth请求的正文。这组参数是登录所需的最低参数。还有其他参数可用。

客户端SRP_A从生成器模数 N g 上升到秘密随机整数 a 的次方。

```
{
   "AuthFlow": "USER_SRP_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "SRP_A" : "[g^a % N]"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito 以 PASSWORD_VERIFIER 质询进行响应。您的客户必须完成 SRP 计算并回应RespondToAuthChallenge或AdminRespondToAuthChallenge请求中的质疑。

```
{
   "ChallengeName": "PASSWORD_VERIFIER",
   "ChallengeResponses": {
      "PASSWORD_CLAIM_SIGNATURE" : "string",
      "PASSWORD_CLAIM_SECRET_BLOCK" : "string",
      "TIMESTAMP" : "string"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

成功响应PASSWORD_VERIFIER质询后，Amazon Cognito 会发放令牌或其他必需的质询，例如多因素身份验证 (MFA)。

使用一次性密码进行无密码登录

密码可能会丢失或被盗。您可能只想验证您的用户是否有权访问经过验证的电子邮件地址、电话号码或身份验证器应用程序。解决这个问题的方法是使用无密码登录。您的应用程序可以提示用户输入其用户名、电子邮件地址或电话号码。然后，Amazon Cognito 会生成一个一次性密码 (OTP)，他们必须确认该密码。成功的代码完成身份验证。

无密码身份验证流程与用户池中必需的多因素身份验证 (MFA) 不兼容。如果 MFA 在您的用户池中是可选的，则激活 MFA 的用户将无法使用无密码第一因素登录。在 MFA 可选用户池中没有 MFA 首选项的用户可以使用无密码因素登录。有关更多信息，请参阅 关于用户池 MFA 的注意事项。

当用户在无密码身份验证中正确输入他们在短信或电子邮件中收到的代码时，除了对用户进行身份验证外，您的用户池还会将该用户的未验证电子邮件地址或电话号码属性标记为已验证。无论您是否UNCONFIRMED将CONFIRMED用户池配置为自动验证电子邮件地址或电话号码，用户状态也从更改为。

支持无密码登录的新选项

当您在用户池中激活无密码身份验证时，它会改变某些用户流程的工作方式。

1. 用户无需密码即可注册，并在登录时选择无密码因素。您也可以以管理员身份创建没有密码的用户。
2. 使用 CSV 文件导入的用户可以立即使用无密码登录。他们无需在登录前设置密码。
3. 没有密码的用户可以提交不带PreviousPassword参数ChangePassword的 API 请求。

使用自动登录 OTPs

通过电子邮件或短信注册并确认其用户帐户的用户 OTPs 可以使用与其确认消息匹配的无密码因素自动登录。在托管登录界面中，确认账户并有资格使用确认码传送方式进行 OTP 登录的用户在提供确认码后会自动进入首次登录。在带有 Amazon SDK 的自定义应用程序中，将以下参数传递给InitiateAuth或AdminInitiateAuth操作。

- ConfirmSignUpAPI 响应中的Session参数作为Session请求参数。
- 其中AuthFlow之一USER_AUTH。

你可以通过EMAIL_OTP或的 PREFER RED_CHALLE NGESMS_OTP，但这不是必需的。该Session参数提供身份验证证明，当您传递有效的会话代码AuthParameters时，Amazon Cognito 会忽略该参数。

登录操作将返回表示身份验证成功的响应，如果满足以下条件 AuthenticationResult，则不会再进行其他质询。

- 该Session代码有效且未过期。
- 用户有资格使用 OTP 身份验证方法。

Activate passwordless sign-in

控制台

要激活无密码登录，请将您的用户池配置为允许使用一种或多种无密码类型进行主登录，然后将您的应用程序客户端配置为允许该流程。USER_AUTH在 Amazon Cognito 控制台中，导航到用户池配置中 “身份验证” 下的 “登录” 菜单。编辑基于选择的登录选项，然后选择通过电子邮件发送一次性密码或短信一次性密码。您可以激活这两个选项。保存您的更改。

导航到应用程序客户端菜单，然后选择应用程序客户端或创建一个新客户端。选择 “编辑”，然后选择 “登录时选择身份验证类型：AL LOW_USER_AUTH”。

API/SDK

在用户池 API 中，`SignInPolicy`使用CreateUserPool或UpdateUserPool请求中的相应无密码选项进行配置。

```
"SignInPolicy": {
    "AllowedFirstAuthFactors": [
        "EMAIL_OTP",
        "SMS_OTP"
    ]
}
```

使用CreateUserPoolClient或UpdateUserPoolClient请求中的必填选项配置您的应用程序客户端`ExplicitAuthFlows`。

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH"
]
```

Sign in with passwordless

无密码登录没有可在和中指定的基于客户端`AuthFlow`的登录方式。InitiateAuthAdminInitiateAuthOTP 身份验证仅在基于选择`AuthFlow`的情况下可用`USER_AUTH`，您可以在其中请求首选登录选项或从用户的登录选项中选择无密码选项。AvailableChallenges要让用户登录应用程序，请按如下方式配置您的`InitiateAuth`或`AdminInitiateAuth`请求的正文。这组参数是登录所需的最低参数。还有其他参数可用。

在此示例中，我们不知道用户想要通过哪种方式登录。如果我们添加一个PREFERRED_CHALLENGE参数并且用户可以使用首选质询，则 Amazon Cognito 会以该质询作为响应。

```
{
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser"
   },
   "ClientId": "1example23456789"
}
```

在此示例AuthParameters中，您可以改"PREFERRED_CHALLENGE": "SMS_OTP"为添加"PREFERRED_CHALLENGE": "EMAIL_OTP"或。如果用户有资格使用该首选方法，则您的用户池会立即向用户的电子邮件地址或电话号码发送验证码，然后返回"ChallengeName": "EMAIL_OTP"或"ChallengeName": "SMS_OTP"。

如果您未指定首选挑战，Amazon Cognito 会使用参数进行AvailableChallenges响应。

```
{
   "AvailableChallenges": [
      "EMAIL_OTP",
      "SMS_OTP",
      "PASSWORD"
    ],
   "Session": "[Session ID]"
}
```

此用户有资格使用电子邮件 OTP、SMS 消息 OTP 和用户名密码进行无密码登录。您的应用程序可以提示用户进行选择，或者根据内部逻辑进行选择。然后，它继续执行选择挑战的RespondToAuthChallenge或AdminRespondToAuthChallenge请求。假设用户想要使用电子邮件消息 OTP 完成无密码身份验证。

```
{
   "ChallengeName": "SELECT_CHALLENGE",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "ANSWER" : "EMAIL_OTP"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
```

```
}
```

Amazon Cognito 以EMAIL_OTP质询作为回应，并将验证码发送到您的用户经过验证的电子邮件地址。然后，您的应用程序必须再次响应此挑战。

如果您以 a 的EMAIL_OTP身份申请，这也将是下一个质询回复PREFERRED_CHALLENGE。

```
{
   "ChallengeName": "EMAIL_OTP",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "EMAIL_OTP_CODE" : "123456"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

使用密钥进行无密码登录 WebAuthn

密钥是安全的，用户的工作量相对较低。Passkey 登录使用身份验证器，即用户可以用来进行身份验证的外部设备。普通密码会让用户面临网络钓鱼、密码猜测和凭据盗窃等漏洞。使用密钥，您的应用程序可以受益于移动电话和其他与信息系统相连或内置于信息系统的设备上的高级安全措施。常见的密钥登录工作流程首先是调用您的设备调用您的密码或凭据管理器，例如 iOS 钥匙串或 Google Chrome 密码管理器。设备端凭证管理器会提示他们选择密钥，并使用现有的凭据或设备解锁机制对其进行授权。现代手机有面部扫描仪、指纹扫描仪、解锁图案和其他机制，有些机制可以同时满足你所知道的东西和你具有强身份验证原则的东西。如果使用生物识别进行密钥身份验证，则密钥代表您的真实身份。

您可能需要将密码替换为指纹、面部或安全密钥身份验证。这是密钥或WebAuthn身份验证。应用程序开发人员通常允许用户在首次使用密码登录后注册生物识别设备。借助 Amazon Cognito 用户池，您的应用程序可以为用户配置此登录选项。密钥身份验证不符合多因素身份验证 (MFA) 的条件。

无密码身份验证流程与用户池中必需的多因素身份验证 (MFA) 不兼容。如果 MFA 在您的用户池中是可选的，则激活 MFA 的用户将无法使用无密码第一因素登录。在 MFA 可选用户池中没有 MFA 首选项的用户可以使用无密码因素登录。有关更多信息，请参阅 关于用户池 MFA 的注意事项。

什么是密钥？

密码无需记住复杂的密码或输入 OTPs，从而简化了用户体验。Passkeys 基于万维网联盟 (W3C) WebAuthn 和 FIDO（Fast Identity Online）联盟起草的 CTAP2 标准。浏览器和平台实施这些标准，提

供 APIs 网络或移动应用程序以启动密钥注册或身份验证过程，还提供用户界面供用户选择密钥身份验证器并与之交互。

当用户在网站或应用程序中注册身份验证器时，身份验证器会创建公私密钥对。 WebAuthn 浏览器和平台将公钥提交给网站或应用程序的应用程序后端。身份验证器保留有关用户和应用程序的私钥 IDs、密钥和元数据。当用户想要使用其注册的身份验证器在注册的应用程序中进行身份验证时，应用程序会生成一个随机质询。对这一质询的回应是使用该应用程序和用户的身份验证器的私钥以及相关的元数据生成的质询数字签名。浏览器或应用程序平台接收数字签名并将其传递到应用程序后端。然后，应用程序使用存储的公钥对签名进行验证。

> **Note**
>
> 您的应用程序不会收到用户向其身份验证器提供的任何身份验证密钥，也不会收到有关私钥的信息。

以下是目前市场上身份验证器的一些示例和功能。身份验证器可能符合其中任何或全部类别。

- 某些身份验证器在授予访问权限之前会使用 PIN、使用面部或指纹输入生物识别信息或密码等因素进行用户验证，从而确保只有合法用户才能授权操作。其他身份验证器没有任何用户验证功能，有些身份验证器可以在应用程序不需要用户验证时跳过用户验证。
- 某些身份验证器（例如 YubiKey 硬件令牌）是便携式的。它们通过 USB、蓝牙或 NFC 连接与设备通信。有些身份验证器是本地的，绑定到某个平台，例如电脑上的 Windows Hello 或 iPhone 上的 Face ID。绑定设备的身份验证器如果足够小，则可以由用户携带，例如移动设备。有时，用户可以通过无线通信将硬件身份验证器与许多不同的平台连接起来。例如，使用桌面浏览器的用户在扫描 QR 码时可以将其智能手机用作密钥身份验证器。
- 一些平台绑定的密钥会同步到云端，因此可以在多个位置使用。例如，iPhone 上的 Face ID 密钥会将密钥元数据与用户的 iCloud 钥匙串中的 Apple 账户同步。这些密钥允许用户在 Apple 设备之间进行无缝身份验证，而不是要求用户单独注册每台设备。1Password、Dashlane 和 Bitwarden 等基于软件的身份验证器应用程序可在用户安装该应用程序的所有平台上同步密钥。

WebAuthn 用术语来说，网站和应用程序是依赖方。每个密钥都与一个特定的依赖方 ID 相关联，这是一个统一的标识符，代表接受密钥身份验证的网站或应用程序。开发人员必须仔细选择其依赖方 ID，才能获得正确的身份验证范围。典型的依赖方 ID 是 Web 服务器的根域名。具有此信赖方 ID 规范的密钥可以对该域和子域进行身份验证。当用户要访问的网站的 URL 与信赖方 ID 不匹配时，浏览器和平台会拒绝密钥身份验证。同样，对于移动应用程序，只有当应用程序在依赖方 ID 所指示的路径上提供的`.well-known`关联文件中存在应用程序路径时，才能使用密钥。

密钥是可以发现的。浏览器或平台可以自动识别和使用它们，而无需用户输入用户名。当用户访问支持密钥身份验证的网站或应用程序时，他们可以从浏览器或平台已经知道的密钥列表中进行选择，也可以扫描二维码。

Amazon Cognito 如何实现密钥身份验证？

Passkeys 是一项可选功能，适用于除精简版以外的所有功能计划。它仅在基于选择的身份验证流程中可用。通过托管登录，Amazon Cognito 可以处理密钥身份验证的逻辑。您还可以使用中的 Amazon Cognito 用户池 API 在 Amazon SDKs应用程序后端进行密钥身份验证。

Amazon Cognito 可以识别使用两种非对称加密算法 (-7) 和 ES256 (-257) 中的任何一种创建的密钥。RS256大多数身份验证器都支持这两种算法。默认情况下，用户可以设置任何类型的身份验证器，例如硬件令牌、移动智能手机和软件身份验证器应用程序。Amazon Cognito 目前不支持认证执法。

在您的用户池中，您可以将用户验证配置为首选或必填项。在不提供值的 API 请求中，此设置默认为首选，Amazon Cognito 控制台中默认选择首选项。当您将用户验证设置为首选时，用户可以设置不具有用户验证功能的身份验证器，并且注册和身份验证操作无需用户验证即可成功。要在密钥注册和身份验证中强制进行用户验证，请将此设置更改为必填项。

您在密钥配置中设置的信赖方 (RP) ID 是一个重要的决定。如果您未另行指定，并且您的域名品牌版本为托管登录，则您的用户池默认为您的自定义域名的名称作为 RP ID。如果您没有自定义域名且未另行指定，则您的用户池默认为您的前缀域名的 RP ID。您也可以将您的 RP ID 配置为不在公共后缀列表 (PSL) 中的任何域名。您的 RP ID 条目适用于托管登录和 SDK 身份验证中的密钥注册和身份验证。Passkey 仅在移动应用程序中起作用，Amazon Cognito 可以找到`.well-known`以您的 RP ID 作为域名的关联文件。作为最佳实践，请在您的网站或应用程序公开之前确定并设置您的信赖方 ID 的值。如果您更改了 RP ID，则您的用户必须使用新的 RP ID 重新注册。

每个用户最多可以注册 20 个密钥。他们只有在至少一次登录您的用户池后才能注册密钥。托管登录可以省去密钥注册的大量工作。当您为用户池和应用程序客户端启用密钥身份验证时，具有托管登录域的用户池会提醒最终用户在注册新用户帐户后注册密钥。您也可以随时调用用户的浏览器，将他们定向到托管登录页面进行密钥注册。在 Amazon Cognito 可以启动密钥身份验证之前，用户必须提供用户名。托管登录会自动处理此问题。登录页面会提示输入用户名，验证用户是否注册了至少一个密钥，然后提示输入密钥登录。同样，基于 SDK 的应用程序必须提示输入用户名并在身份验证请求中提供该用户名。

当您使用密钥设置用户池身份验证并且拥有自定义域和前缀域时，RP ID 默认为自定义域的完全限定域名 (FQDN)。要在 Amazon Cognito 控制台中将前缀域设置为 RP ID，请删除您的自定义域名或将前缀域的 FQDN 作为第三方域名输入。

Activate passkey sign-in

控制台

要激活使用密钥登录，请将您的用户池配置为允许使用一种或多种无密码类型进行主登录，然后将您的应用程序客户端配置为允许该流程。USER_AUTH在 Amazon Cognito 控制台中，导航到用户池配置中 “身份验证” 下的 “登录” 菜单。编辑基于选择的登录的选项，然后将 Passkey 添加到可用选项列表中。

导航到身份验证方法菜单并编辑密钥。

- 用户验证是用于确定您的用户池是否需要密钥设备来执行额外检查当前用户是否已获得密钥授权的密钥的设置。要鼓励用户为设备配置用户验证但不要求用户验证，请选择 “首选”。要仅支持带有用户验证功能的设备，请选择 “必需”。有关更多信息，请参阅 w3. org 上的用户验证。
- 信赖方 ID 的域是您的应用程序将在用户的密钥注册请求中传递的标识符。它设定了与用户密钥颁发者建立信任关系的目标。您的信赖方 ID 可以是：您的用户池的域，如果

  Cognito 域

  您的用户池中的 Amazon Cognito 前缀域。

  CUSTOM 域

  您的用户池的自定义域。

  第三方域名

  不使用用户池托管登录页面的应用程序的域。此设置通常与没有域的用户池相关联，这些用户池在后端使用 S Amazon DK 和用户池 API 进行身份验证。

导航到应用程序客户端菜单，然后选择应用程序客户端或创建一个新客户端。选择 “编辑”，然后在 “身份验证流程” 下选择 “登录时选择身份验证类型：AL LOW_USER_AUTH”。

API/SDK

在用户池 API 中，`SignInPolicy`使用CreateUserPool或UpdateUserPool请求中的相应密钥选项进行配置。密钥身份验证`WEB_AUTHN`选项必须至少附带一个其他选项。密钥注册需要现有的身份验证会话。

```
"SignInPolicy": {
    "AllowedFirstAuthFactors": [
        "PASSWORD",
        "WEB_AUTHN"
```

```
    ]
}
```

在SetUserPoolMfaConfig请求的`WebAuthnConfiguration`参数中配置您的用户验证首选项和 RP ID。是`RelyingPartyId`密钥身份验证结果的预期目标，可以是您的用户池前缀或自定义域，也可以是您自己选择的域。

```
"WebAuthnConfiguration": {
   "RelyingPartyId": "example.auth.us-east-1.amazoncognito.com",
   "UserVerification": "preferred"
}
```

使用CreateUserPoolClient或UpdateUserPoolClient请求中的必填选项配置您的应用程序客户端`ExplicitAuthFlows`。

```
"ExplicitAuthFlows": [
   "ALLOW_USER_AUTH"
]
```

Register a passkey (managed login)

托管登录处理密钥的用户注册。当您的用户池中激活密钥身份验证时，Amazon Cognito 会在用户注册新用户账户时提示用户设置密钥。

当用户已经注册但未设置密钥或您以管理员身份创建账户时，Amazon Cognito 不会提示他们设置密钥。处于这种状态的用户必须使用其他因素（例如密码或无密码 OTP）登录，然后才能注册密钥。

注册密钥

1. 将用户引导至您的登录页面。

   ```
   https://auth.example.com/oauth2/authorize/?
   client_id=1example23456789&response_type=code&scope=email+openid
   +phone&redirect_uri=https%3A%2F%2Fwww.example.com
   ```

2. 处理用户的身份验证结果。在此示例中，Amazon Cognito 使用您的应用程序交换令牌`www.example.com`的授权码将其重定向到。
3. 将用户引导至您的注册密码页面。用户将拥有一个用于维护其登录会话的浏览器 Cookie。密钥 URL 需要`client_id`和`redirect_uri`参数。Amazon Cognito 仅允许经过身份验证的用户

访问此页面。使用密码、电子邮件 OTP 或 SMS OTP 登录您的用户，然后调用符合以下模式的 URL。

您也可以在此请求中添加其他对端点授权参数，例如response_type和scope。

```
https://auth.example.com/passkeys/add?
client_id=1example23456789&redirect_uri=https%3A%2F%2Fwww.example.com
```

Register a passkey (SDK)

您可以在PublicKeyCreationOptions对象中使用元数据注册密钥凭证。您可以使用登录用户的凭据生成此对象，然后在 API 请求中将其呈现给他们的密钥颁发者。颁发者将返回一个确认密钥注册的 RegistrationResponseJSON 对象。

要开始密钥注册过程，请使用现有登录选项登录用户。使用当前用户的访问令牌授权StartWebAuthnRegistration的 API 请求进行授权。以下是示例GetWebAuthnRegistrationOptions请求的正文。

```
{
   "AccessToken": "eyJra456defEXAMPLE"
}
```

来自用户池的响应包含PublicKeyCreationOptions对象。在 API 请求中向用户的颁发者展示此对象。它提供诸如公钥和依赖方 ID 之类的信息。颁发者将使用RegistrationResponseJSON对象进行回应。

在 CompleteWebAuthnRegistrationAPI 请求中显示注册响应，再次使用用户的访问令牌进行授权。当您的用户池使用空正文的 HTTP 200 响应进行响应时，您的用户密钥已被注册。

Sign in with a passkey

无密码登录没有您可以在和AuthFlow中指定的选项。InitiateAuthAdminInitiateAuth相反，您必须声明 o AuthFlow USER_AUTH f 并请求登录选项，或者从用户池的响应中选择您的无密码选项。要让用户登录应用程序，请按如下方式配置您的InitiateAuth或AdminInitiateAuth请求的正文。这组参数是登录所需的最低参数。还有其他参数可用。

在此示例中，我们知道用户想要使用密钥登录，因此我们添加了一个PREFERRED_CHALLENGE参数。

```
{
```

```
   "AuthFlow": "USER_AUTH",
   "AuthParameters": {
      "USERNAME" : "testuser",
      "PREFERRED_CHALLENGE" : "WEB_AUTHN"
   },
   "ClientId": "1example23456789"
}
```

Amazon Cognito 以 WEB_AUTHN 质询进行响应。您的应用程序必须应对这一挑战。向用户的密钥提供者发起登录请求。它将返回一个 AuthenticationResponseJSON 对象。

```
{
   "ChallengeName": "WEB_AUTHN",
   "ChallengeResponses": {
      "USERNAME" : "testuser",
      "CREDENTIAL" : "{AuthenticationResponseJSON}"
   },
   "ClientId": "1example23456789",
   "Session": "[Session ID from the previous response]"
}
```

登录后的 MFA

您可以设置通过用户名-密码流程完成登录的用户，系统会提示他们使用电子邮件、SMS 消息或代码生成应用程序中的一次性密码进行额外验证。MFA 不同于无密码登录，后者是使用一次性 WebAuthn 密码或不包含 MFA 的密钥的第一个身份验证因素。用户池中的 MFA 是一种质询-响应模型，即用户首先证明自己知道密码，然后证明自己有权访问注册的第二要素设备。

实施资源

- 向用户池添加 MFA

刷新令牌

当您想让用户保持登录状态而不重新输入其凭据时，刷新令牌是您的应用程序保留用户会话所必需的工具。应用程序可以向您的用户池提供刷新令牌，并将其交换为新的 ID 和访问令牌。通过令牌刷新，您可以确保登录用户仍处于活动状态，获取更新的属性信息，并在无需用户干预的情况下更新访问控制权限。

实施资源

- 刷新令牌

自定义身份验证

您可能需要为用户配置一种未在此处列出的身份验证方法。您可以使用带有 Lambda 触发器的自定义身份验证来做到这一点。在一系列 Lambda 函数中，Amazon Cognito 发出质疑，询问用户必须回答的问题，检查答案是否准确，然后确定是否应发出另一个质询。问题和答案可以包括安全问题、对 CAPTCHA 服务的请求、对外部 MFA 服务 API 的请求或所有这些按顺序排列。

实施资源

- 自定义身份验证质询 Lambda 触发器

自定义身份验证流程

Amazon Cognito 用户池实现了使用自定义身份验证流程，这可以帮助您使用 Amazon Lambda 触发器创建基于质询/响应的身份验证模型。

自定义身份验证流程可以自定义质询和响应周期，以满足不同需求。该流程首先调用 `InitiateAuth` API 操作，该调用指示将使用的身份验证类型并提供了所有初始身份验证参数。Amazon Cognito 将使用以下类型的信息之一响应 `InitiateAuth` 调用：

- 用户质询及会话和参数。
- 错误（如果用户未能通过身份验证）
- ID、访问和刷新令牌（如果 `InitiateAuth` 调用中提供的参数足以使用户登录）。（通常，用户或应用程序必须首先应答质询，但这必须由您的自定义代码决定。）

如果 Amazon Cognito 使用质询响应 `InitiateAuth` 调用，则应用程序将收集更多输入并调用 `RespondToAuthChallenge` 操作。此调用提供质询响应并将其传回会话。Amazon Cognito 对 `RespondToAuthChallenge` 的响应类似于对 `InitiateAuth` 调用的响应。如果用户已登录，Amazon Cognito 会提供令牌，如果用户未登录，则 Amazon Cognito 会提供另一个质询或错误。如果 Amazon Cognito 返回另一质询，则序列重复，应用程序调用 `RespondToAuthChallenge` 直到用户成功登录或返回错误。有关 `InitiateAuth` 和 `RespondToAuthChallenge` API 操作的详细信息，请参阅 API 文档。

自定义身份验证流程和质询

应用程序可以启动自定义身份验证流程，具体方法是：调用 InitiateAuth 并将 CUSTOM_AUTH 用作 Authflow。借助自定义身份验证流程，三个 Lambda 触发器控制响应的质询和验证。

- DefineAuthChallenge Lambda 触发器将以前的质询和响应的会话数组作为输入。然后，它生成下一个质询名称和布尔值，指示用户是否通过身份验证并且应被授予令牌。此 Lambda 触发器是一个状态机，可通过质询控制用户的路径。
- CreateAuthChallenge Lambda 触发器将质询名称作为输入并生成质询和参数以评估响应。当 DefineAuthChallenge 返回 CUSTOM_CHALLENGE 作为下一次质询时，身份验证流程调用 CreateAuthChallenge。CreateAuthChallenge Lambda 触发器在质询元数据参数中传递下一个类型的质询。
- VerifyAuthChallengeResponse Lambda 函数会评估响应并返回布尔值以表明响应是否有效。

自定义身份验证流程还可以使用内置质询的组合，例如 SRP 密码验证和通过短信进行的 MFA。它可以使用自定义质询，如验证码或秘密问题。

在自定义身份验证流程中使用 SRP 密码验证

如果您希望将 SRP 包含在自定义身份验证流程中，则您必须开始使用 SRP。

- 要在自定义流程中启动 SRP 密码验证，应用程序将 CUSTOM_AUTH 作为 Authflow 来调用 InitiateAuth。在 AuthParameters 映射 中，来自应用程序的请求包括 SRP_A:（SRP A 值）和 CHALLENGE_NAME: SRP_A。
- CUSTOM_AUTH 流会使用 challengeName: SRP_A 和 challengeResult: true 的初始会话调用 DefineAuthChallenge Lambda 触发器。您的 Lambda 函数使用 challengeName: PASSWORD_VERIFIER、issueTokens: false 和 failAuthentication: false 作出响应。
- 接下来，该应用程序必须使用 challengeName: PASSWORD_VERIFIER 和 challengeResponses 映射中 SRP 所需的其它参数调用 RespondToAuthChallenge。
- 如果 Amazon Cognito 验证了密码，RespondToAuthChallenge 使用 challengeName: PASSWORD_VERIFIER 和 challengeResult: true 的第二个会话调用 DefineAuthChallenge Lambda 触发器。此时，DefineAuthChallenge Lambda 触发器可以使用 challengeName: CUSTOM_CHALLENGE 响应来开启自定义质询。
- 如果为用户启用了 MFA，则在 Amazon Cognito 验证密码后，您的用户将被要求设置 MFA 或使用 MFA 登录。

> **Note**
>
> Amazon Cognito 托管的登录网页无法激活 自定义身份验证质询 Lambda 触发器。

有关 Lambda 触发器的更多信息，包括示例代码，请参阅使用 Lambda 触发器自定义用户池工作流。

用户迁移身份验证流程

用户迁移 Lambda 触发器可帮助您将用户从旧式用户管理系统迁移到您的用户池。如果选择 `USER_PASSWORD_AUTH` 身份验证流程，则用户在用户迁移过程中无需重置密码。此流程在身份验证期间通过加密的 SSL 连接向服务发送用户的密码。

所有用户均完成迁移后，请切换为更安全的 SRP 流程。SRP 流程不通过网络发送任何密码。

要了解有关 Lambda 触发器的更多信息，请参阅使用 Lambda 触发器自定义用户池工作流。

有关使用 Lambda 触发器迁移用户的更多信息，请参阅利用用户迁移 Lambda 触发器导入用户。

API 和 SDK 身份验证的授权模型

当您开始使用用户池身份验证开发应用程序时，必须决定适合您的应用程序类型的 API 授权模型。授权模型是一种系统，用于通过 Amazon Cognito 用户池 API 和 SDK 集成中的身份验证组件提供请求的授权。Amazon Cognito 有三种授权模式：IAM 授权、公开和令牌授权。

对于 IAM 授权的请求，授权来自请求`Authorization`标题中一组 Amazon IAM 证书的签名。对于服务器端应用程序，这种做法通过 IAM 授权保护身份验证操作。对于公开（未经身份验证的）身份验证请求，无需授权。这适用于分发给用户的客户端应用程序。对于令牌授权的操作（通常与公共操作结合实施），授权来自`Authorization`请求标头中包含的会话令牌或访问令牌。Amazon Cognito 身份验证通常要求您按顺序实现两个或更多 API 操作，而您使用的 API 操作取决于您的应用程序的特征。公共客户端（将应用程序分发给用户）使用公共操作，而登录请求不需要授权。令牌授权的操作继续使用公共应用程序中的用户会话。服务器端客户端（应用程序逻辑托管在远程系统上）通过 IAM 对登录请求的授权来保护身份验证操作。接下来的 API 操作对及其相应的 SDK 方法映射到可用的授权模型。

每个公共身份验证操作都有某种形式的服务器端等效操作，例如UpdateUserAttributes和。AdminUpdateUserAttributes虽然客户端操作是由用户启动的，需要确认，但服务器端操作假定更改是由用户池管理员提交的，并且更改会立即生效。在此示例中，Amazon Cognito 向用户发送了一条包含确认码的消息，用户的访问令牌授权了提交验证码的VerifyUserAttribute请求。服务器端应用程序可以立即设置任何属性的值，但在使用电子邮件地址和电话号码进行登录时，需要特别注意更改电子邮件地址和电话号码的值。

要比较 API 身份验证并查看 API 操作及其授权模型的完整列表，请参阅了解 API、OIDC 和托管登录页面身份验证。

Client-side (public) authentication

以下是客户端应用程序中的典型请求序列

1. 公共InitiateAuth操作会提交主要证书，例如用户名和密码。
2. 令牌授权的RespondToAuthChallenge操作从`InitiateAuth`响应和质询的答案中提交会话令牌，例如 MFA。会话令牌授权表示属于 not-yet-complete身份验证周期一部分的请求。
3. 令牌授权ConfirmDevice操作提交访问令牌并执行写入操作，将记住的设备添加到用户的个人资料中。访问令牌授权表示用户在完成身份验证后提出的自助服务操作请求。

有关更多信息，请参阅客户端身份验证选项和了解 API、OIDC 和托管登录页面身份验证。

Server-side authentication

以下是来自服务器端操作的典型请求序列。每个请求都有一个Amazon 签名版本 4 授权标头，使用颁发给应用程序服务器的 IAM 计算机证书进行签名。

1. 该AdminInitiateAuth操作会提交主要凭证，例如用户名和密码。
2. AdminRespondToAuthChallenge操作提交质疑的答案，例如 MFA。
3. 该AdminUpdateDeviceStatus操作将`AdminInitiateAuth`响应中的设备密钥设置为已记住的。

有关更多信息，请参阅服务器端身份验证选项和了解 API、OIDC 和托管登录页面身份验证。

用户通过应答连续的质询进行身份验证，直到身份验证失败或 Amazon Cognito 向用户发放令牌。您可以在包含不同质询的流程中对 Amazon Cognito 重复这些步骤，以支持任何自定义身份验证流程。

主题

- 服务器端身份验证选项
- 客户端身份验证选项
- 了解 API、OIDC 和托管登录页面身份验证
- 按授权模型分组的 API 操作列表

服务器端身份验证选项

Web 应用程序和其他服务器端应用程序在远程服务器上实现身份验证，客户端将其加载到远程显示应用程序（如浏览器或 SSH 会话）中。服务器端应用程序通常具有以下特征。

- 它们是在服务器上安装的 Java、Ruby 或 Node.js 等语言的应用程序中构建的。
- 它们连接到可能有客户端密钥的用户池应用程序客户端，称为机密客户端。
- 他们有权访问 Amazon 证书。
- 他们调用托管登录进行身份验证，或者在用户池 API 中使用 IAM 授权的操作和 SD Amazon K。
- 他们为内部客户提供服务，也可能为公共客户提供服务。

使用用户池 API 的服务器端操作可以使用密码、一次性密码或密钥作为主要登录因素。对于服务器端应用程序，用户池身份验证与客户端应用程序的身份验证类似，但以下情况除外：

- 服务器端应用程序发出 AdminInitiateAuthAPI 请求。此操作需要具有包含`cognito-idp:AdminInitiateAuth`和权限的 Amazon 证书`cognito-idp:AdminRespondToAuthChallenge`。该操作返回所需的质询或身份验证结果。
- 当应用程序收到质询时，它会发出 AdminRespondToAuthChallengeAPI 请求。`AdminRespondToAuthChallenge`API 操作还需要 Amazon 凭证。

有关使用 Amazon 凭证签署 Amazon Cognito API 请求的更多信息，请参阅Amazon 一般参考中的签名版本 4 签名流程。

在 `AdminInitiateAuth` 响应 `ChallengeParameters` 中，`USER_ID_FOR_SRP` 属性（如果存在）包含用户的实际用户名而不是别名（如电子邮件地址或电话号码）。在 `AdminRespondToAuthChallenge` 调用的 `ChallengeResponses` 中，您必须在 `USERNAME` 参数中传递此用户名。

> **Note**
>
> 由于后端管理员实现使用管理员身份验证流程，因此该流程不支持记住的设备。在您启用设备跟踪时，管理员身份验证成功，但任何对刷新访问令牌的调用均会失败。

客户端身份验证选项

移动应用程序和其他客户端应用程序类型安装在用户的设备上，并在本地执行身份验证和用户界面逻辑。它们通常具有以下特征。

- 它们采用 React native、Flutter 和 Swift 等语言构建，并部署到用户设备上。
- 它们连接到没有客户端密钥的用户池应用程序客户端，称为公共客户端。
- 他们无法访问授权 IAM 授权的 API 请求的 Amazon 证书。
- 他们调用托管登录进行身份验证，或者使用 SDK 在用户池 API 中使用公开和令牌授权的操作。Amazon
- 他们为公众客户提供服务，并允许任何人注册和登录。

使用用户池 API 的客户端操作可以使用密码、一次性密码或密钥作为主要登录因素。以下过程适用于您使用Amazon Amplify或创建的用户客户端应用程序。Amazon SDKs

1. 用户将他们的用户名和密码输入到应用程序中。
2. 应用程序使用用户的用户名和安全远程密码（SRP）详细信息调用 `InitiateAuth` 操作。

   此 API 操作返回身份验证参数。

   > **Note**
   > 该应用程序使用内置的 Amazon Cognito SRP 功能生成 SRP 详细信息。 Amazon SDKs

3. 应用程序调用 `RespondToAuthChallenge` 操作。如果调用成功，则 Amazon Cognito 返回用户的令牌，并且身份验证流程完成。

   如果 Amazon Cognito 需要另一个质询，则对 `RespondToAuthChallenge` 的调用不返回任何令牌。相反，调用会返回一个会话。
4. 如果 `RespondToAuthChallenge` 返回一个会话，应用程序将再次调用 `RespondToAuthChallenge`，这次使用会话和质询响应（例如，MFA 代码）。

了解 API、OIDC 和托管登录页面身份验证

Amazon Cognito 用户池是多种身份验证技术的组合。他们是外部身份提供商 (IdPs) 的依赖方。它们 IdPs 适用于使用 OpenID Connect (OIDC) 实现身份验证的应用程序。 SDKs它们作为 JSON Web 令

牌 (JWTs) 的发行者提供身份验证，类似于 OIDC 身份验证，但使用的是其中的一部分 API 方法。Amazon SDKs它们也可以是应用程序的安全入口点。

当您想要注册、登录和管理用户池中的用户时，有两种选择。

1. 您的托管登录页面和经典托管用户界面包括托管登录用户交互式端点以及处理 IdP 和 r elying-party 角色的联合终端节点。它们构成了一个公共网页包，当您为用户池选择域时，Amazon Cognito 会激活这些网页。要快速开始使用 Amazon Cognito 用户池的身份验证和授权功能，包括注册、登录、密码管理和多重身份验证 (MFA) 页面，请使用托管登录的内置用户界面。

   其他用户池端点便于使用第三方身份提供商进行身份验证（IdPs）。他们执行的服务包括以下各项。

   a. 服务提供商回调终端节点，用于来自您的经过身份验证的声明 IdPs，比如`saml2/idpresponse`和。`oauth2/idpresponse`当 Amazon Cognito 是您的应用程序和 IdP 之间的中间服务提供者（SP）时，回调端点代表服务。

   b. 提供有关您的环境的信息的端点，例如 `oauth2/userInfo` 和 `/.well-known/jwks.json`。您的应用在使用 OIDC 或 2.0 开发者库验证令牌或检索用户个人资料数据时会使用这些端点。OAuth

2. Amazon Cognito 用户池 API 是一组工具，供您的网络或移动应用程序在您自己的自定义前端收集登录信息后对用户进行身份验证。用户池 API 身份验证生成以下 JSON Web 令牌。

   a. 带有来自您的用户的可验证属性声明的身份令牌。

   b. 一种访问令牌，用于授权用户针对 Amazon 服务端点创建经令牌授权的 API 请求。

   > **Note**
   >
   > 默认情况下，来自用户池 API 身份验证的访问令牌仅包含 `aws.cognito.signin.user.admin` 作用域。要生成具有额外作用域的访问令牌（例如，授权对第三方 API 的请求），请在通过用户池端点进行身份验证期间请求作用域，或者在 令牌生成前 Lambda 触发器 中添加自定义作用域。自定义访问令牌会增加 Amazon 账单费用。

   c. 一种刷新令牌，用于授权请求新 ID 和访问令牌，并刷新用户身份和访问控制属性。

您可以将通常通过用户池端点登录的联合用户与其配置文件位于用户池本地 的用户相关联。本地用户仅存在于您的用户池目录中，无需通过外部 IdP 进行联合身份验证。如果您在 AdminLinkProviderForUserAPI 请求中将他们的联合身份关联到本地用户，则他们可以使用用户池 API 登录。有关更多信息，请参阅 将联合用户与现有用户配置文件关联。

Amazon Cognito 用户池 API 有双重用途。

1. 它创建和配置您的 Amazon Cognito 用户池资源。例如，您可以创建用户池、添加 Amazon Lambda 触发器以及配置托管您的托管登录页面的用户池域。
2. 它为本地用户和关联用户执行注册、登录和其他用户操作。

使用 Amazon Cognito 用户池 API 的示例场景

1. 用户选择了您在应用程序中创建的“创建账户”按钮。他们输入电子邮件地址和密码。
2. 您的应用程序发送 SignUpAPI 请求并在您的用户池中创建了一个新用户。
3. 应用程序提示用户输入电子邮件确认代码。用户输入他们在电子邮件中收到的代码。
4. 您的应用会发送包含用户确认码的 ConfirmSignUpAPI 请求。
5. 应用程序提示您的用户输入用户名和密码，而用户输入其信息。
6. 您的应用程序发送 InitiateAuthAPI 请求并存储 ID 令牌、访问令牌和刷新令牌。应用程序调用 OIDC 库来管理用户的令牌并为该用户维护持久会话。

在 Amazon Cognito 用户池 API 中，您无法登录通过 IdP 进行联合身份验证的用户。您必须通过用户池端点对这些用户进行身份验证。有关包含托管登录的用户池终端节点的更多信息，请参阅用户池端点和托管登录参考。

您的联合用户可以从托管登录开始并选择他们的 IdP，也可以跳过托管登录，将您的用户直接发送到您的 IdP 进行登录。当您的 API 发送请求到 对端点授权 且带有 IdP 参数时，Amazon Cognito 会以静默方式将用户重定向到 IdP 登录页面。

托管登录页面的示例场景

1. 用户选择了您在应用程序中创建的“创建账户”按钮。
2. 托管登录会向您的用户显示您在其中注册开发者凭证的社交身份提供商的列表。您的用户选择了 Apple。
3. 您的应用程序向 对端点授权 发出请求，提供者名称为 `SignInWithApple`。
4. 用户的浏览器会打开 Apple 身份验证页面。您的用户登录并选择授权 Amazon Cognito 阅读他们的个人资料信息。
5. Amazon Cognito 确认 Apple 访问令牌并查询用户的 Apple 个人资料。
6. 用户向您的应用程序出示 Amazon Cognito 授权代码。

7. 应用程序中的 OIDC 库与交换授权码，令牌端点并存储用户池发布的 ID 令牌、访问令牌和刷新令牌。您的应用程序使用 OIDC 库来管理用户的令牌并为该用户维护持续会话。

用户池 API 和托管登录页面支持本指南中描述的各种场景。以下部分探讨了用户池 API 如何进一步划分为支持您的注册、登录和资源管理要求的类。

按授权模型分组的 API 操作列表

Amazon Cognito 用户池 API 既是资源管理接口，也是面向用户的身份验证和授权接口，结合了其操作中遵循的授权模型。根据 API 操作，您可能需要使用 IAM 凭证、访问令牌、会话令牌、客户端密钥或者前面这些内容的组合提供授权。对于许多用户身份验证和授权操作，您可以选择请求的经过身份验证和未经身份验证的版本。对于分发给用户的应用程序（例如移动应用程序），最佳安全实践是提供未经身份验证的操作；您无需在代码中包含任何密钥。

您只能在 IAM policy 中为 IAM 授权的管理操作 和 IAM 授权的用户操作 分配权限。

IAM 授权的管理操作

IAM 授权的管理操作会修改和查看您的用户池和应用程序客户端配置，就像您在中所做的那样。Amazon Web Services Management Console

例如，要在 UpdateUserPoolAPI 请求中修改您的用户池，您必须出示 Amazon 证书和 IAM 权限才能更新资源。

要在 Amazon Command Line Interface (Amazon CLI) 或 Amazon 软件开发工具包中授权这些请求，请使用环境变量或向请求添加 IAM 证书的客户端配置来配置您的环境。有关更多信息，请参阅中的Amazon 使用您的 Amazon 凭证进行访问Amazon Web Services 一般参考。对于 Amazon Cognito 用户池 API，您也可以直接向服务端点发送请求。您必须使用在请求标题中嵌入的 Amazon 凭据对这些请求进行授权或签名。有关更多信息，请参阅签署 Amazon API 请求。

IAM 授权的管理操作

AddCustomAttributes

CreateGroup

CreateIdentityProvider

CreateResourceServer

IAM 授权的管理操作

CreateUserImportJob

CreateUserPool

CreateUserPoolClient

CreateUserPoolDomain

DeleteGroup

DeleteIdentityProvider

DeleteResourceServer

DeleteUserPool

DeleteUserPoolClient

DeleteUserPoolDomain

DescribeIdentityProvider

DescribeResourceServer

DescribeRiskConfiguration

DescribeUserImportJob

DescribeUserPool

DescribeUserPoolClient

DescribeUserPoolDomain

获取 CSVHeader

GetGroup

GetIdentityProviderByIdentifier

GetSigningCertificate

IAM 授权的管理操作

获取 UICustomization

GetUserPoolMfaConfig

ListGroups

ListIdentityProviders

ListResourceServers

ListTagsForResource

ListUserImportJobs

ListUserPoolClients

ListUserPools

ListUsers

ListUsersInGroup

SetRiskConfiguration

Set UICustomization

SetUserPoolMfaConfig

StartUserImportJob

StopUserImportJob

TagResource

UntagResource

UpdateGroup

UpdateIdentityProvider

UpdateResourceServer

IAM 授权的管理操作

UpdateUserPool

UpdateUserPoolClient

UpdateUserPoolDomain

IAM 授权的用户操作

IAM 授权用户操作注册、登录、管理用户证书、修改和查看您的用户。

例如，您可以有一个服务器端应用程序层，为 Web 前端提供支持。您的服务器端应用程序是您信任的 OAuth 机密客户端，可以访问您的 Amazon Cognito 资源。要在应用程序中注册用户，您的服务器可以在 AdminCreateUserAPI 请求中包含 Amazon 凭据。有关 OAuth 客户端类型的更多信息，请参阅 OAuth 2.0 授权框架中的客户端类型。

要在 Amazon CLI 或 Amazon SDK 中授权这些请求，请使用环境变量或向请求添加 IAM 证书的客户端配置来配置服务器端应用程序环境。有关更多信息，请参阅中的Amazon 使用您的 Amazon 凭证进行访问Amazon Web Services 一般参考。对于 Amazon Cognito 用户池 API，您也可以直接向服务端点发送请求。您必须使用在请求标题中嵌入的 Amazon 凭据对这些请求进行授权或签名。有关更多信息，请参阅签署 Amazon API 请求。

如果您的应用程序客户端有客户端密钥，则您必须提供 IAM 凭证，并在 `AuthParameters` 中提供 `SecretHash` 参数或 `SECRET_HASH` 值（取决于操作）。有关更多信息，请参阅 计算密钥哈希值。

IAM 授权的用户操作

AdminAddUserToGroup

AdminConfirmSignUp

AdminCreateUser

AdminDeleteUser

AdminDeleteUserAttributes

AdminDisableProviderForUser

IAM 授权的用户操作

AdminDisableUser

AdminEnableUser

AdminForgetDevice

AdminGetDevice

AdminGetUser

AdminInitiateAuth

AdminLinkProviderForUser

AdminListDevices

AdminListGroupsForUser

AdminListUserAuthEvents

AdminRemoveUserFromGroup

AdminResetUserPassword

AdminRespondToAuthChallenge

AdminSetUserMFAPreference

AdminSetUserPassword

AdminSetUserSettings

AdminUpdateAuthEventFeedback

AdminUpdateDeviceStatus

AdminUpdateUserAttributes

AdminUserGlobalSignOut

未经身份验证的用户操作

未经身份验证的用户操作注册、登录以及为您的用户启动密码重置。当您希望 Internet 上的任何人都可以注册并登录您的应用程序时，请使用未经身份验证（公开）的 API 操作。

例如，要在您的应用程序中注册用户，您可以分发一个不提供任何密钥特权访问权限的 OAuth 公共客户端。您可以使用未经身份验证的 API 操作SignUp注册此用户。

要在使用 Amazon SDK 开发的公共客户端中发送这些请求，您无需配置任何凭据。对于没有额外授权的 Amazon Cognito 用户池 API，您也可以直接向服务端点发送请求。

如果您的应用程序客户端有客户端密钥，则您必须在 `AuthParameters` 中提供 `SecretHash` 参数或 `SECRET_HASH` 值（取决于操作）。有关更多信息，请参阅 计算密钥哈希值。

未经身份验证的用户操作

SignUp

ConfirmSignUp

ResendConfirmationCode

ForgotPassword

ConfirmForgotPassword

InitiateAuth

经过令牌授权的用户操作

经过令牌授权的用户操作在用户已登录或开始登录流程之后，注销、管理其凭证、修改和查看用户。如果您不想在应用程序中分发密钥，并且想要使用用户自己的凭证授权请求，请使用经过令牌授权的 API 操作。如果用户已完成登录，则您必须使用访问令牌来授权其经过令牌授权的 API 请求。如果您的用户正在登录流程中，则您必须使用 Amazon Cognito 在对先前请求的响应中返回的会话令牌，授权其经过令牌授权的 API 请求。

例如，在公共客户端中，您可能希望更新用户的配置文件，限制用户仅对自己的配置文件具有写入权限。要进行此更新，您的客户端可以在 UpdateUserAttributesAPI 请求中包含用户的访问令牌。

要在使用 Amazon SDK 开发的公共客户端中发送这些请求，您无需配置任何凭据。在您的请求中包含 `AccessToken` 或 `Session` 参数。对于 Amazon Cognito 用户池 API，您也可以直接向服务端点发送请求。要向服务端点授权请求，请在请求的 POST 正文中包含访问令牌或会话令牌。

要签署经过令牌授权的操作的 API 请求，请将访问令牌作为 `Authorization` 标头包含在请求中，格式为 `Bearer <Base64-encoded access token>`。

经过令牌授权的用户操作	AccessToken	会话
RespondToAuthChallenge		✓
ChangePassword	✓	
GetUser	✓	
StartWebAuthnRegistration	✓	
CompleteWebAuthnRegistration	✓	
DeleteWebAuthnCredential	✓	
ListWebAuthnCredentials	✓	
UpdateUserAttributes	✓	
DeleteUserAttributes	✓	
DeleteUser	✓	
ConfirmDevice	✓	
ForgetDevice	✓	
GetDevice	✓	

经过令牌授权的用户操作	AccessToken	会话
ListDevices	✓	
UpdateDeviceStatus	✓	
GetUserAttributeVerificationCode	✓	
VerifyUserAttribute	✓	
SetUserSettings	✓	
SetUserMFAPreference	✓	
GlobalSignOut	✓	
UpdateAuthEventFeedback		✓
AssociateSoftwareToken	✓	✓
VerifySoftwareToken	✓	✓
RevokeToken[1]		
GetTokensFromRefreshToken[1]		

[1] `RevokeToken` `GetTokensFromRefreshToken` 并将刷新令牌作为授权参数。刷新令牌用作授权令牌和目标资源。

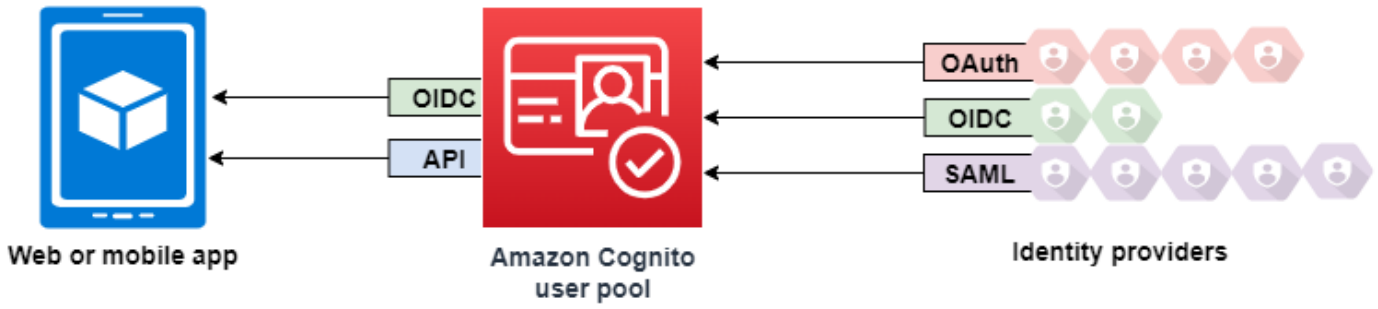
用户池使用第三方身份提供商登录

您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供者（IdP）进行联合身份验证。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs借助内置的托管网络用户界面，Amazon Cognito

为所有经过身份验证的用户提供令牌处理和管理。 IdPs这样，后端系统可以基于一组用户池令牌实现标准化。

联合登录在 Amazon Cognito 用户群体中的工作方式

通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此特征不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。



Amazon Cognito 是一个用户目录和 OAuth 2.0 身份提供商 (IdP)。当您以本地用户身份登录 Amazon Cognito 目录时，您的用户群体是应用程序的 IdP。本地用户仅存在于您的用户池目录中，无需通过外部 IdP 进行联合身份验证。

当你将 Amazon Cognito 连接到社交、SAML 或 OpenID Connect (OIDC IdPs) 时，你的用户池充当了多个服务提供商和你的应用程序之间的桥梁。对于您的 IdP 而言，Amazon Cognito 是服务提供商 (SP)。你将 OIDC ID 令牌或 SAML 声明 IdPs 传递给 Amazon Cognito。Amazon Cognito 会在令牌或断言中读取有关您用户的声明，并将这些声明映射到用户群体目录中的新用户配置文件。

然后，Amazon Cognito 在其自己的目录中为联合用户创建用户配置文件。Amazon Cognito 根据来自 IdP 的声明向用户添加属性。对于 OIDC 和社交身份提供者，则向 IDP 运营的公有 `userinfo` 端点添加属性。当映射的 IdP 属性发生变化时，用户的属性会在用户群体中发生变化。您还可以添加更多属性，这些属性独立于 IdP 中的属性。

Amazon Cognito 为联合用户创建配置文件后，它会更改其功能并将自己显示为应用程序的 IdP（现在是 SP）。Amazon Cognito 是 OIDC 和 2.0 IdP 的组合。 OAuth 它生成访问令牌、ID 令牌和刷新令牌。有关令牌的更多信息，请参阅了解用户池 JSON 网络令牌 (JWTs)。

您必须设计一个与 Amazon Cognito 集成的应用程序，以便对用户进行身份验证和授权，无论是联合用户还是本地用户。

应用程序作为 Amazon Cognito 的服务提供者的责任

验证和处理令牌中的信息

在大多数情况下，Amazon Cognito 将经过身份验证的用户重新导向到它附加了授权码的应用程序 URL。您的应用程序将此代码交换为访问权限、ID 和刷新令牌。然后，它必须检查令牌的有效性，并根据令牌中的声明向用户提供信息。

使用 Amazon Cognito API 请求响应身份验证事件

您的应用程序必须与 Amazon Cognito 用户群体 API 和身份验证 API 端点集成。身份验证 API 会将用户进行登录和注销，并管理令牌。用户群体 API 具有多种操作，用于管理您的用户群体、用户以及身份验证环境的安全性。当应用程序收到来自 Amazon Cognito 的响应时，它必须知道下一步该怎么做。

关于 Amazon Cognito 用户群体第三方登录需要了解的事项

- 如果您希望用户使用联合提供商登录，则必须选择域。这将为托管登录设置页面。有关更多信息，请参阅 使用自己的域名进行托管登录。
- 您无法使用 InitiateAuth和之类的 API 操作登录联合用户AdminInitiateAuth。联合用户只能使用登录端点或对端点授权进行登录。
- 对端点授权 是重定向端点。如果您在请求中提供`idp_identifier`或`identity_provider`参数，它会绕过托管登录，静默重定向到您的 IdP。否则，它会重定向到托管登录登录端点。
- 当托管登录将会话重定向到联合身份提供商时，Amazon Cognito 会在请求`user-agent`中`Amazon/Cognito`包含标头。
- Amazon Cognito 从固定标识符和 IdP 名称的组合中派生联合用户配置文件的 `username` 属性。要生成符合自定义要求的用户名，请创建到 `preferred_username` 属性的映射。有关更多信息，请参阅 有关映射的需知信息。

  示例：`MyIDP_bob@example.com`
- Amazon Cognito 会为您添加到用户池中的每个 OIDC SAMl 和社交 IdP 创建一个用户组。群组名称的格式为`[user pool ID]_[IdP name]`，例如`us-east-1_EXAMPLE_MYSSO`或`us-east-1_EXAMPLE_Google`。自动生成的每个唯一的 IdP 用户配置文件都会自动添加到该群组中。关联用户不会自动添加到该群组，但您可以通过单独的流程将其个人资料添加到群组。
- Amazon Cognito 将有关联合用户身份的信息记录到属性中，并在 ID 令牌中记录一个称为 `identities` 的声明。此声明包含用户的提供商以及提供商提供的唯一 ID。您无法直接在用户配置

文件中更改 `identities` 属性。有关如何关联联合用户的更多信息，请参阅将联合用户与现有用户配置文件关联。

- 当您在 UpdateIdentityProviderAPI 请求中更新 IdP 时，您的更改最多可能需要一分钟才能显示在托管登录中。
- Amazon Cognito 支持在其自身与您的 IdP 之间最多 20 个 HTTP 重定向。
- 当您的用户使用托管登录登录时，他们的浏览器会存储一个加密的登录会话 Cookie，用于记录他们登录的客户端和提供商。如果他们尝试使用相同的参数再次登录，则托管登录会重复使用任何未过期的现有会话，并且用户无需再次提供凭据即可进行身份验证。如果用户使用不同的 IdP 再次登录，包括切换到本地用户群体登录或从本地用户群体登录进行切换，则他们必须提供凭证并生成新的登录会话。

  您可以将任何用户池分配 IdPs 给任何应用程序客户端，并且用户只能使用您分配给其应用程序客户端的 IdP 登录。

主题

- 为用户池配置身份提供商
- 将社交身份提供者与用户池配合使用
- 将 SAML 身份提供者与用户池配合使用
- 将 OIDC 身份提供者与用户池配合使用
- 将 IdP 属性映射到配置文件和令牌
- 将联合用户与现有用户配置文件关联

为用户池配置身份提供商

使用用户池，您可以通过各种外部身份提供商（IdPs）实现登录。指南的这一部分说明了如何在 Amazon Cognito 控制台中使用用户池设置这些身份提供者。或者，您可以使用用户池 API 和 Amazon SDK 以编程方式添加用户池身份提供商。有关更多信息，请参阅 CreateIdentityProvider。

支持的身份提供者选项包括 Facebook、Google 和 Amazon 等社交提供者，以及 OpenID Connect（OIDC）和 SAML 2.0 提供者。在开始之前，请为自己设置 IdP 的管理凭证。对于每种类型的提供者，您都需要注册应用程序，获取必要的凭证，然后在用户池中配置提供者的详细信息。然后，您的用户可以使用已连接身份提供者的现有账户注册和登录您的应用程序。

“身份验证” 下的 “社交和外部提供商” 菜单可添加和更新用户池 IdPs。有关更多信息，请参阅 用户池使用第三方身份提供商登录。

主题

- 使用社交 IdP 设置用户登录
- 使用 OIDC IdP 设置用户登录
- 使用 SAML IdP 设置用户登录

使用社交 IdP 设置用户登录

您可以使用联合身份验证，将 Amazon Cognito 用户池与社交身份提供商（如 Facebook、Google 和 Login with Amazon）集成起来。

要添加社交身份提供商，您首先要通过该身份提供商创建一个开发人员账户。在拥有开发人员账户后，您应向身份提供商注册您的应用程序。该身份提供商将为您的应用程序创建应用程序 ID 和应用程序密钥，然后您在您的 Amazon Cognito 用户池中配置这些值。

- Google Identity Platform
- Facebook for Developers
- Login with Amazon
- 通过 Apple 登录

将用户登录与社交 IdP 集成

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 选择 “社交和外部提供商” 菜单。
4. 选择 Add an identity provider（添加身份提供商），或者选择已配置的身份提供商（例如 Facebook、Google、Amazon 或 Apple），找到 Identity provider information（身份提供商信息），然后选择 Edit（编辑）。有关添加社交身份提供商的更多信息，请参阅将社交身份提供者与用户池配合使用。
5. 根据您选择的 IdP，完成以下步骤之一，从而输入社交身份提供商的信息：

   Facebook、Google 和 Login with Amazon

   输入您创建客户端应用程序时收到的应用程序密钥。

Sign In with Apple

输入您向 Apple 提供的服务 ID，以及创建应用程序客户端时收到的团队 ID、密钥 ID 和私有密钥。

6. 对于 Authorize scopes（授权范围），输入要映射到用户池属性的社交身份提供商范围的名称。范围定义了您要通过应用程序访问的用户属性（如名称和电子邮件）。输入范围时，根据您选择的 IdP 使用以下准则：

   - Facebook – 以英文逗号分隔范围。例如：

     `public_profile, email`

   - Google、Login with Amazon 和 Sign In with Apple – 以空格分隔范围。例如：
     - Google: `profile email openid`
     - Login with Amazon: `profile postal_code`
     - 通过 Apple 登录：`name email`

     > **Note**
     > 对于 Sign in with Apple（控制台），请使用复选框以选择范围。

7. 选择保存更改。
8. 从 “应用程序客户端” 菜单中，从列表中选择应用程序客户端，然后选择 “编辑”。将新的社交身份提供商添加到 Identity providers（身份提供商）下的应用程序客户端。
9. 选择保存更改。

有关社交的更多信息 IdPs，请参阅将社交身份提供者与用户池配合使用。

使用 OIDC IdP 设置用户登录

您可以将用户登录与 OpenID Connect (OIDC) 身份提供商 (IdP) 集成，例如 Salesforce 或 Ping Identity。

向用户群体添加 OIDC 提供者

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 从导航菜单中选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。

4. 选择 “社交和外部提供商” 菜单，然后选择 “添加身份提供商”。
5. 选择 OpenID Connect 身份提供商。
6. 在 Provider name（提供商名称）中输入一个唯一名称。
7. 将您从提供商那里收到的客户端 ID 输入到 Client ID（客户端 ID）。
8. 将您从提供商那里收到的客户端密钥输入到 Client secret（客户端密钥）。
9. 为该提供商输入 Authorized scopes（授权范围）。范围定义了应用程序将向您的提供商请求的用户属性组（例如 `name` 和 `email`）。按照 OAuth 2.0 规范，作用域必须用空格分隔。

   您的用户必须同意向您的应用程序提供这些属性。
10. 请选择一个 Attribute request method（属性请求方法），以便为 Amazon Cognito 提供 HTTP 方法（GET 或 POST），Amazon Cognito 使用该方法从提供商运营的 userInfo 端点中获取用户的详细信息。
11. 请选择 Setup method（设置方法）并通过 Auto fill through issuer URL（自动填充发布者 URL）或 Manual input（手动输入）检索 OpenID Connect 端点。如果您的提供商拥有公共`.well-known/openid-configuration`终端节点，Amazon Cognito 可以在其中检索、、和`jwks_uri`终端节点 `authorization` `tokenuserInfo`，请 URLs 使用自动填写发行者 URL。
12. 输入 URLs 来自您的 IdP 的发卡机构 URL 或`authorizationtokenuserInfo`、、和`jwks_uri`终端节点。

    > **Note**
    >
    > 您只能使用端口号 443 和 80 进行发现、自动填充和手动输入。 URLs如果您的 OIDC 提供商使用任何非标准 TCP 端口，则用户登录失败。
    > 发布者 URL 必须以 `https://` 开头，而且不得以 `/` 字符结尾。例如，Salesforce 使用以下 URL：
    > `https://login.salesforce.com`
    > 与您的发卡机构 URL 关联的`openid-configuration`文档必须 URLs 为以下值提供 HTTPS：`authorization_endpoint``token_endpoint`、`userinfo_endpoint`、和`jwks_uri`。同样，当您选择 “手动输入” 时，只能输入 HTTPS URLs。
13. 默认情况下，sub OIDC 声明将映射到用户池 Username（用户名）属性中。您可以将其他 OIDC 声明映射到用户池属性。输入 OIDC 声明，然后从下拉列表中选择对应的用户池属性。例如，声明 email 通常会映射到用户池属性 Email（电子邮件）。

14. 请将身份提供商的其它属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
15. 选择创建。
16. 从应用程序客户端菜单中，从列表中选择一个应用程序客户端，然后选择编辑。要将新的 SAML 身份提供商添加到应用程序客户端，请导航到登录页面选项卡，然后选择在托管登录页面配置上编辑。
17. 选择保存更改。

有关 OIDC 的更多信息 IdPs，请参阅。将 OIDC 身份提供者与用户池配合使用

使用 SAML IdP 设置用户登录

您可以使用 Amazon Cognito 用户池的联合身份验证与 SAML 身份提供商（IdP）集成。您可以通过上传文件或输入元数据文档端点 URL 来提供元数据文档。有关获取第三方 SAML 的元数据文档的信息 IdPs，请参阅配置第三方 SAML 身份提供者。

在您的用户池中配置 SAML 2.0 身份提供商

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “社交和外部提供商” 菜单，然后选择 “添加身份提供者”。
5. 选择 SAML 身份提供商。
6. 输入以逗号分隔的 Identifiers（标识符）。标识符指示 Amazon Cognito 检查用户登录电子邮件地址，然后将用户引导到与其域对应的提供商。
7. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。配置您的 SAML 2.0 身份提供商，使其向 Amazon Cognito 在配置托管登录时创建的`https://mydomain.auth.us-east-1.amazoncognito.com/saml2/logout`终端节点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

> **Note**
>
> 如果选择此选项，并且您的 SAML 身份提供商需要已签名的注销请求，则您还需要为您的 SAML IdP 配置 Amazon Cognito 提供的签名证书。
> SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。

8. 选择 Metadata document source（元数据文档源）。如果您的身份提供商在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

> **Note**
>
> 如果您的提供商具有公有端点，我们建议您输入元数据文档 URL，而不是上载文件。如果您使用 URL，Amazon Cognito 会自动刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。

9. Map attributes between your SAML provider and your app（在 SAML 提供商和应用程序之间映射属性）将 SAML 提供程序属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

   例如，当您选择 User pool attribute（用户池属性）`email` 时，按照您的身份提供商提供的 SAML 断言中显示的内容，输入 SAML 属性名称。您的身份提供商可能会提供示例 SAML 断言以供参考。一些身份提供商使用简单名称（如 `email`），另一些则使用类似于下面 URL 格式的属性名称：

   ```
   http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
   ```

10. 选择 Create（创建）。

> **Note**
>
> 如果您在使用 HTTPS 元数据端点 URL 创建 SAML IdP 时看见 `InvalidParameterException`，请确保元数据端点已正确设置 SSL，并且存在与之关联的有效 SSL 证书。此类异常的一个例子是 “从中检索元数据时出错*<metadata endpoint>*”。

设置 SAML IdP 以添加签名证书

- 要获取包含 IdP 用来验证已签名注销请求的公钥的证书，请执行以下操作：
  1. 转到用户池的 “社交和外部提供商” 菜单。
  2. 选择您的 SAML 提供商，
  3. 选择查看签名证书。

有关 SAML 的更多信息， IdPs 请参阅将 SAML 身份提供者与用户池配合使用。

将社交身份提供者与用户池配合使用

您的 Web 和移动应用程序用户可以通过社交身份提供商 (IdP)（例如 Facebook、Google、Amazon 和 Apple）进行登录。利用内置托管 Web UI，Amazon Cognito 将为所有经过身份验证的用户提供令牌处理和管理。这样，后端系统可以基于一组用户池令牌实现标准化。您必须启用托管登录才能与支持的社交身份提供商集成。当 Amazon Cognito 构建您的托管登录页面时，它会创建 OAuth 2.0 个终端节点，Amazon Cognito 和您的 OIDC 以及 IdPs 社交使用这些终端节点来交换信息。有关更多信息，请参阅 Amazon Cognito 用户池 Auth API 参考。

你可以在中添加社交 IdP Amazon Web Services Management Console，也可以使用 CLI 或 Amazon Cognito AP Amazon I。

> **Note**
>
> 通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此特征不依赖于通过 Amazon Cognito 身份池（联合身份）实现的联合身份验证。

主题

- 设置社交 IdP 开发者账户和应用程序
- 使用社交 IdP 配置您的用户池
- 测试社交 IdP 配置

设置社交 IdP 开发者账户和应用程序

在使用 Amazon Cognito 创建社交 IdP 之前，必须向社交 IdP 注册应用程序才能接收客户端 ID 和客户端密钥。

Facebook

有关 Meta 开发者账户配置和身份验证的最新信息，请参阅 Meta 应用程序开发。

如何在 Facebook/Meta 上注册应用程序

1. 创建 Facebook 开发人员账户。
2. 使用 Facebook 凭证登录。
3. 在 My Apps（我的应用程序）菜单上，选择 Create New App（创建新的应用程序）。

4. 输入 Facebook 应用程序的名称，然后选择 Create App ID（创建应用程序 ID）。
5. 在左侧导航栏上，选择 Settings（设置），然后选择 Basic（基本）。
6. 记下 App ID（应用程序 ID）和 App Secret（应用程序密钥）。您将在下一节中使用它们。
7. 从页面底部选择 + Add Platform（+ 添加平台）。
8. 选择 Website（网站）。
9. 在 Website（网站）下，请在您的应用程序登录页面上将路径输入到 Site URL（站点 URL）中。

```
https://mydomain.auth.us-east-1.amazoncognito.com/login?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

10. 选择 Save changes（保存更改）。
11. 在您的用户池域的根目录中，将路径输入到 App Domains（应用程序域）。

```
https://mydomain.auth.us-east-1.amazoncognito.com
```

12. 选择 Save changes（保存更改）。
13. 从导航栏中，选择 Add Product（添加产品），然后选择 Facebook Login（Facebook 登录）中的 Set up（设置）。
14. 从导航栏中，选择 Facebook Login（Facebook 登录），然后选择 Settings（设置）。

    在 “有效 OAuth重定向” 中输入您的用户池域的`/oauth2/idpresponse`终端节点路径 URIs。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

15. 选择保存更改。

Login with Amazon

有关 Login with Amazon 开发者账户配置和身份验证的最新信息，请参阅 L ogin with Amazon 文档。

如何使用 Login with Amazon 注册应用程序

1. 创建 Amazon 开发人员账户。

2. 使用 Amazon 凭证登录。
3. 您需要创建一个 Amazon 安全配置文件才能接收 Amazon 客户端 ID 和客户端密钥。

   从页面顶部的导航栏中选择 Apps and Services（应用程序和服务），然后选择 Login with Amazon。
4. 选择 Create a Security Profile（创建安全配置文件）。
5. 输入 Security Profile Name（安全配置文件名称）、Security Profile Description（安全配置文件描述）和 Consent Privacy Notice URL（同意隐私声明 URL）。
6. 选择 Save（保存）。
7. 选择 Client ID（客户端 ID）和 Client Secret（客户端密钥）以显示客户端 ID 和密钥。您将在下一节中使用它们。
8. 将鼠标悬停在齿轮图标上并选择 Web Settings（Web 设置），然后选择 Edit（编辑）。
9. 将用户池域输入到 Allowed Origins（允许的源）中。

   ```
   https://mydomain.auth.us-east-1.amazoncognito.com
   ```
10. 在 “允许的返回” 中输入您的用户池域和`/oauth2/idpresponse`终端节点 URLs。

    ```
    https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```
11. 选择保存。

Google

有关 Google Cloud 平台中 OAuth 2.0 的更多信息，请参阅 Google Workspace 开发者版文档中的了解身份验证和授权。

如何向谷歌注册应用程序

1. 创建 Google 开发人员账户。
2. 登录到 Google Cloud Platform 控制台。
3. 从顶部导航栏中，选择 Select a project（选择项目）。如果您在 Google 平台中已经有项目，则此菜单会改为显示您的默认项目。
4. 选择 NEW PROJECT（新建项目）。
5. 输入产品的名称，然后选择 CREATE（创建）。
6. 在左侧导航栏上，选择APIs 和服务，然后选择 Oauth 同意屏幕。

7. 输入应用程序信息，App domain（应用程序域）、Authorized domains（已授权的域）和 Developer contact information（开发人员联系信息）。例如，您的 Authorized domains（已授权的域）必须包括 `amazoncognito.com` 和自定义域的根，例如 `example.com`。选择 SAVE AND CONTINUE（保存并继续）。
8. 1. 在 “范围” 下，选择 “添加或移除范围”，然后至少选择以下 OAuth 范围。

   1. `.../auth/userinfo.email`
   2. `.../auth/userinfo.profile`
   3. openid
9. 在 Test users（测试用户）下，选择 Add users（添加用户）。输入您的电子邮件地址和任何其他授权测试用户，然后选择 SAVE AND CONTINUE（保存并继续）。
10. 再次展开左侧导航栏，选择APIs 和服务，然后选择凭证。
11. 选择创建凭据，然后选择OAuth 客户端 ID。
12. 选择 Application type（应用程序类型）并为客户端提供 Name（名称）。
13. 在 “授权 JavaScript 来源” 下，选择 “添加 URI”。输入用户群体域。

    ```
    https://mydomain.auth.us-east-1.amazoncognito.com
    ```

14. 在 “授权重定向” 下 URIs，选择 “添加 URI”。输入指向用户群体域的 `/oauth2/idpresponse` 端点的路径。

    ```
    https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
    ```

15. 选择 CREATE（创建）。
16. 安全地存储 Google 在 Your client ID（您的客户端 ID）和 Your client secret（您的客户端密钥）下显示的值。当您添加 Google IdP 时，请向 Amazon Cognito 提供这些值。

Sign in with Apple

有关设置 “使用 Apple 登录” 的 up-to-date更多信息，请参阅 Apple 开发者文档中的配置环境以使用 Apple 登录。

如何通过 “用 Apple 登录” (SIWA) 注册应用程序

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。

3. 在左侧导航栏上，选择 Certificates, Identifiers & Profiles（证书、标识符和配置文件）。
4. 在左侧导航栏上，选择 Identifiers (标识符)。
5. 在 Identifiers (标识符) 页面上，选择 + 图标。
6. 在 “注册新标识符” 页面上，选择 “应用程序” IDs，然后选择 “继续”。
7. 在 Select a type（选择类型）页面上，选择 App（应用程序），然后选择 Continue（继续）。
8. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：
   1. 在 Description（说明）下，输入说明。
   2. 在 App ID Prefix（应用程序 ID 前缀）下，输入 Bundle ID（捆绑包 ID）。记下 App ID Prefix（应用程序 ID 前缀）下的值。在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供商后，您将使用此值。
   3. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。
   4. 在 “使用 Apple 登录：应用程序 ID 配置” 页面上，选择将应用程序设置为主应用程序或与其他应用程序分组 IDs，然后选择 “保存”。
   5. 选择 Continue（继续）。
9. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。
10. 在 Identifiers (标识符) 页面上，选择 + 图标。
11. 在 “注册新标识符” 页上，选择 “服务” IDs，然后选择 “继续”。
12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：
    1. 在 Description（描述）下方，键入描述。
    2. 在 Identifier（标识符）下方，键入标识符。记下此服务 ID，因为在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供商后需要此值。
    3. 选择 Continue（继续），然后选择 Register（注册）。
13. 从 Identifiers（标识符）页中选择您刚创建的 Services ID（服务 ID）。
    1. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。
    2. 在 Web Authentication Configuration（Web 身份验证配置）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。
    3. 选择 “网站” 旁边的 “+” 图标 URLs。
    4. 在 Domains and subdomains（域名和子域）下，输入不带 `https://` 前缀的用户群体域。

```
mydomain.auth.us-east-1.amazoncognito.com
```

5. 在 Return 下 URLs，输入您的用户池域/oauth2/idpresponse终端节点的路径。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

6. 选择 Next（下一步），然后选择 Done（完成）。您不需要验证域。
7. 选择 Continue（继续），然后选择 Save（保存）。

14. 在左侧导航栏上，选择 Keys (密钥)。
15. 在 Keys（密钥）页面上，选择 + 图标。
16. 在 Register a New Key（注册新密钥）页面上，执行以下操作：

    1. 在 Key Name（密钥名称）下，输入密钥名称。
    2. 选择 Sign In with Apple，然后选择 Configure（配置）。
    3. 在 Configure Key（配置密钥）页上，选择您先前创建的应用程序 ID 作为 Primary App ID（主应用程序 ID）。选择保存。
    4. 选择 Continue（继续），然后选择 Register（注册）。
17. 在 Download Your Key（下载您的密钥）页面上，选择 Download（下载）以下载私有密钥并记下显示的 Key ID（密钥 ID），然后选择 Done（完成）。在使用社交 IdP 配置您的用户池中选择 Apple 作为身份提供商后，您将需要此私有密钥和在此页面上显示的 Key ID（密钥 ID）值。

使用社交 IdP 配置您的用户池

要配置用户池社交 IdP，请使用 Amazon Web Services Management Console

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择现有用户池或创建用户池。
4. 选择 “社交和外部提供商” 菜单，然后选择 “添加身份提供商”。
5. 选择一个社交 IdP：Facebook、Google（谷歌）、Login with Amazon（使用亚马逊账号登录）或 Sign in with Apple（使用苹果账号登录）。
6. 根据您选择的社交 IdP，从以下步骤中进行选择：

    - Google 和 Login with Amazon – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥）。

- Facebook – 输入在上一部分中生成的 app client ID（应用程序客户端 ID）和 app client secret（应用程序客户端密钥），然后选择 API 版本（例如，版本 2.12）。我们建议选择最新的可用版本，因为每个 Facebook API 版本都有一个生命周期和一个弃用日期。Facebook 的范围和属性可能因 API 版本而异。我们建议您使用 Facebook 测试您的社交身份登录，以确保联合身份认证会按预期运行。
- Sign In with Apple – 输入在上一部分中生成的 Services ID（服务 ID）、Team ID（团队 ID）、Key ID（密钥 ID）和 private key（私有密钥）。

7. 输入要使用的 Authorized scopes（授权范围）的名称。范围定义了您要通过应用程序访问的用户属性（如 `name` 和 `email`）。对于 Facebook，这些属性应用逗号分隔。对于 Google 和 Login with Amazon，则应采用空格分隔。对于 Sign in with Apple，选中要访问的范围的复选框。

社交身份提供商	示例范围
Facebook	`public_profile, email`
Google	`profile email openid`
Login with Amazon	`profile postal_code`
Sign in with Apple	`email name`

您的应用程序用户需要同意向您的应用程序提供这些属性。关于社交服务提供商范围的更多信息，请参阅 Google、Facebook 和 Login with Amazon 或 Sign in with Apple 的文档。

对于 Sign in with Apple，下面提供了可能不会返回范围的用户场景：

- 终端用户离开 Apple 登录页面后出现故障（可能来自 Amazon Cognito 内部的故障或开发人员编写的任何内容）
- 服务 ID 标识符可在不同用户池和/或其他身份验证服务中使用
- 在最终用户登录之前，开发人员添加了其他范围（未检索到新信息）
- 开发人员删除用户，然后用户再次登录，而没有从其 Apple ID 配置文件中删除该应用程序

8. 请将 IdP 的属性映射到您的用户池。有关更多信息，请参阅指定适用于用户池的身份提供程序属性映射。
9. 选择创建。

10. 从 “应用程序客户端” 菜单中，从列表中选择应用程序客户端，然后选择 “编辑”。要将新的社交身份提供者添加到应用程序客户端，请导航到登录页面选项卡，然后选择在托管登录页面配置上编辑。
11. 选择保存更改。

测试社交 IdP 配置

在您的应用程序中，您必须在用户的客户端中调用浏览器，这样他们才能使用社交服务提供商登录。完成前几节中的设置过程后，请使用社交服务提供商测试登录情况。以下示例网址会加载带有前缀域名的用户池的登录页面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

当您进入应用程序客户端菜单，选择应用程序客户端，导航到登录页面选项卡，然后选择查看登录页面时，Amazon Cognito 会引导您进入该页面。有关用户池域的更多信息，请参阅 配置用户池域。有关应用程序客户端（包括客户端 IDs 和回调）的更多信息 URLs，请参阅特定于应用程序的应用程序客户端设置。

以下示例链接对端点授权使用`identity_provider`查询参数设置了从中静默重定向到社交提供商。此 URL 绕过使用托管登录的交互式用户池登录，直接转到 IdP 登录页面。

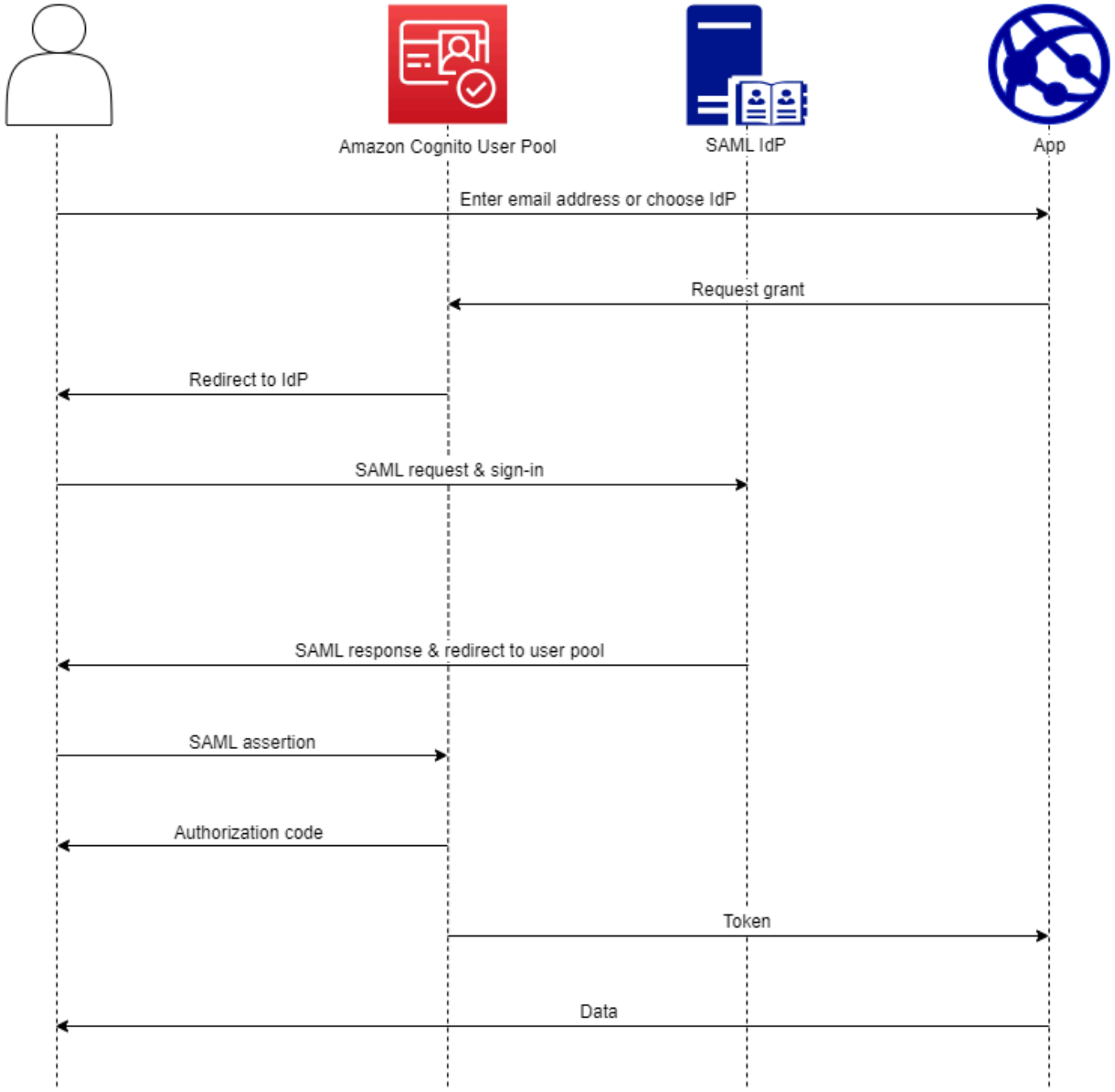
```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/
authorize?identity_provider=Facebook|Google|LoginWithAmazon|
SignInWithApple&response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

将 SAML 身份提供者与用户池配合使用

可以选择允许 Web 和移动应用程序用户通过 SAML 身份提供者（IdP）[如 Microsoft Active Directory 联合身份验证服务（ADFS）或 Shibboleth] 登录。您必须选择支持 SAML 2.0 标准的 SAML IdP。

通过托管登录，Amazon Cognito 可以对本地和第三方 IdP 用户进行身份验证，并发放 JSON 网络令牌 ()。JWTs使用 Amazon Cognito 发放的令牌，您可以将多个身份源整合为适用于所有应用程序的通用 OpenID Connect（OIDC）标准。Amazon Cognito 可以处理来自第三方提供者的 SAML 断言，并将其转换为 SSO 标准。您可以在、通过 Amazon CLI或使用 Amazon Cognito 用户池 API 创建和管理

SAML IdP。 Amazon Web Services Management Console要在中创建您的第一个 SAML IdP Amazon Web Services Management Console，请参阅。在用户池中添加和力 SAML 身份提供者



> **Note**
>
> 通过第三方 IdP 登录进行联合身份验证是 Amazon Cognito 用户池的一项特征。Amazon Cognito 身份池（有时称为 Amazon Cognito 联合身份）是一种联合身份验证的实现，您必须

在每个身份池中单独进行设置。用户池可以是身份池的第三方 IdP。有关更多信息，请参阅 Amazon Cognito 身份池。

IdP 配置的快速参考

您必须配置 SAML IdP，以便接受请求并向用户池发送响应。SAML IdP 的文档将包含相关信息，让您可以了解如何将用户池添加为 SAML 2.0 IdP 的依赖方或应用程序。接下来的文档将提供必须为 SP 实体 ID 和断言使用者服务（ACS）URL 提供的值。

用户池 SAML 值快速参考

SP 实体 ID

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

ACS URL

```
https://Your user pool domain/saml2/idpresponse
```

您必须配置用户池来支持您的身份提供者。添加外部 SAML IdP 的大致步骤如下。

1. 从您的 IdP 下载 SAML 元数据，或检索元数据端点的 URL。请参阅配置第三方 SAML 身份提供者。
2. 将新的 IdP 添加到用户池。上传 SAML 元数据或提供元数据 URL。请参阅在用户池中添加和力 SAML 身份提供者。
3. 将 IdP 分配给您的应用程序客户端。请参阅特定于应用程序的应用程序客户端设置。

主题

- 关于 Amazon Cognito 用户 IdPs 池中 SAML 的注意事项
- SAML 用户名区分大小写
- 配置第三方 SAML 身份提供者
- 在用户池中添加和力 SAML 身份提供者
- SAML 会话在 Amazon Cognito 用户池中启动
- 通过单点注销来注销 SAML 用户
- SAML 签名和加密

- SAML 身份提供者名称和标识符

关于 Amazon Cognito 用户 IdPs 池中 SAML 的注意事项

实施 SAML 2.0 IdP 有一些要求和限制。实施 IdP 时，请参阅此部分。您还将找到一些有用的信息，可用于排查对用户池进行 SAML 联合身份验证期间的错误。

Amazon Cognito 会为您处理 SAML 断言

Amazon Cognito 用户群体支持 SAML 2.0 与 POST 绑定端点联合身份验证。这使您的应用程序不必检索或分析 SAML 断言响应，因为用户池直接通过用户代理从 IdP 接收 SAML 响应。您的用户池代表您的应用程序充当服务提供商（SP）。Amazon Cognito 支持 SP 发起和 IdP 发起的单点登录（SSO），如 SAML V2.0 技术概览的第 5.1.2 节和第 5.1.4 节中所述。

提供有效的 IdP 签名证书

在用户池中配置 SAML IdP 时，SAML 提供者元数据中的签名证书不得过期。

用户池支持多个签名证书

如果在 SAML 元数据中，您的 SAML IdP 包含多个签名证书，则在登录时，只要与 SAML 元数据中的任何证书匹配，您的用户群体就会确定 SAML 断言有效。每份签名证书的长度不得超过 4096 个字符。

维护中继状态参数

Amazon Cognito 和您的 SAML IdP 使用 `relayState` 参数维护会话信息。

1. Amazon Cognito 支持大于 80 个字节的 `relayState` 值。虽然 SAML 规范规定 `relayState` 值“长度不得超过 80 个字节”，但目前的行业惯例往往偏离这种行为。因此，拒绝超过 80 个字节的 `relayState` 值将破坏许多标准 SAML 提供商集成。
2. `relayState` 令牌是对 Amazon Cognito 维护的状态信息的不透明引用。Amazon Cognito 不保证 `relayState` 参数的内容。不要解析其内容，以免您的应用程序依赖解析结果。有关更多信息，请参阅 SAML 2.0 规范。

识别 ACS 端点

您的 SAML 身份提供者要求您设置断言使用者端点。您的 IdP 使用 SAML 断言将您的用户重定向到此端点。在用户群体域中为您的 SAML 身份提供者中的 SAML 2.0 POST 绑定配置以下端点。

```
https://Your user pool domain/saml2/idpresponse
With an Amazon Cognito domain:
https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse
With a custom domain:
```

```
https://auth.example.com/saml2/idpresponse
```

有关用户群体域的更多信息，请参阅 配置用户池域。

没有重播的断言

您无法向您的 Amazon Cognito `saml2/idpresponse` 端点重复或重放 SAML 断言。重放的 SAML 断言的断言 ID 与早期 IdP 响应的 ID 重复。

用户池 ID 是 SP 实体 ID

您必须向 IdP 提供服务提供商（SP）`urn` 中的用户池 ID，也称为受众 URI 或 SP 实体 ID。用户群体的受众 URI 采用以下格式。

```
urn:amazon:cognito:sp:us-east-1_EXAMPLE
```

您可以在 Amazon Cognito 控制台的用户池概览下找到用户池 ID。

映射所有必需属性

配置 SAML IdP，为用户群体中根据需要设置的任何属性提供值。例如，`email` 是用户群体的通用必需属性。在您的用户可以登录之前，SAML IdP 断言必须包含映射到用户群体属性 `email` 的声明。有关属性映射的更多信息，请参阅将 IdP 属性映射到配置文件和令牌。

断言格式有特定的要求

SAML IdP 必须在 SAML 断言中包括以下声明。

- `NameID` 声明。Amazon Cognito 通过 `NameID` 将 SAML 断言与目标用户关联起来。如果 `NameID` 发生变化，Amazon Cognito 会认为这是针对新用户的断言。您在 IdP 配置中设置为 `NameID` 的属性必须具有永久值。要将 SAML 用户分配给用户池中一致的用户配置文件，请根据一个值不变的属性分配您的 `NameID` 声明。

  ```
  <saml2:NameID Format="urn:oasis:names:tc:SAML:1.1:nameid-format:persistent">
    carlos
  </saml2:NameID>
  ```

  `urn:oasis:names:tc:SAML:1.1:nameid-format:persistent` 的 IdP `NameID` 声明中的 `Format` 表示您的 IdP 正在传递一个不变的值。Amazon Cognito 不需要这种格式声明，如果您的 IdP 没有指定 `NameID` 声明的格式，则会分配 `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified` 格式。此行为符合 SAML 2.0 规范的复杂类型名称IDType第 2.2. 2 节。
- 一项 `AudienceRestriction` 声明，所具有的 `Audience` 值将您的用户群体 SP 实体 ID 设置为响应的目标。

```
<saml:AudienceRestriction>
  <saml:Audience> urn:amazon:cognito:sp:us-east-1_EXAMPLE
</saml:AudienceRestriction>
```

- 对于 SP 发起的单点登录，`Response` 元素具有原始 SAML 请求 ID 的 `InResponseTo` 值。

```
<saml2p:Response Destination="https://mydomain.auth.us-east-1.amazoncognito.com/
saml2/idpresponse" ID="id123" InResponseTo="_dd0a3436-bc64-4679-
a0c2-cb4454f04184" IssueInstant="Date-time stamp" Version="2.0"
 xmlns:saml2p="urn:oasis:names:tc:SAML:2.0:protocol" xmlns:xs="http://
www.w3.org/2001/XMLSchema">
```

> **Note**
>
> IdP 发起的 SAML 断言不得包含 `InResponseTo` 值。

- 一个 `SubjectConfirmationData` 元素，具有用户池 `saml2/idpresponse` 端点的 `Recipient` 值，而对于 SP 发起的 SAML，具有与原始 SAML 请求 ID 匹配的 `InResponseTo` 值。

```
<saml2:SubjectConfirmationData InResponseTo="_dd0a3436-bc64-4679-a0c2-
cb4454f04184" NotOnOrAfter="Date-time stamp" Recipient="https://mydomain.auth.us-
east-1.amazoncognito.com/saml2/idpresponse"/>
```

SP 发起的登录请求

当 对端点授权 将用户定向到您的 IdP 登录页面时，Amazon Cognito 会在 HTTP GET 请求的 URL 参数中包括 SAML 请求。SAML 请求包含有关用户池（包括 ACS 端点）的信息。您可以选择对这些请求应用加密签名。

签署请求和加密响应

每个使用 SAML 提供者的用户池都会生成一个非对称密钥对和签名证书，以便让 Amazon Cognito 将数字签名分配给 SAML 请求。您配置为支持加密 SAML 响应的每个外部 SAML IdP 都会导致 Amazon Cognito 为该提供者生成新的密钥对和加密证书。要查看和下载带有公钥的证书，请在 Amazon Cognito 控制台的 “社交和外部提供商” 菜单中选择您的 IdP。

要与来自用户池的 SAML 请求建立信任，可以向 IdP 提供用户池 SAML 2.0 签名证书的副本。如果您未将 IdP 配置为接受已签名的请求，则您的 IdP 可能会忽略您的用户池签署的 SAML 请求。

1. Amazon Cognito 将数字签名应用于用户传递给 IdP 的 SAML 请求。您的用户池签署所有单点注销（SLO）请求，您可以将用户池配置为签署任何 SAML 外部 IdP 的单点登录（SSO）请求。当您提供证书副本时，您的 IdP 会验证用户的 SAML 请求的完整性。
2. 您的 SAML IdP 可以使用加密证书来加密 SAML 响应。当您配置采用 SAML 加密的 IdP 时，您的 IdP 只能发送加密的响应。

对非字母数字字符进行编码

Amazon Cognito 不接受您的 IdP 作为属性值传递的 4 字节 UTF-8 字符（例如 # 或 #）。您可以将字符编码为 Base64，将其作为文本传递，然后在应用程序中对其进行解码。

在以下示例中，将不接受属性声明：

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">#</saml2:AttributeValue>
</saml2:Attribute>
```

与上述示例不同，将接受以下属性声明：

```
<saml2:Attribute Name="Name" NameFormat="urn:oasis:names:tc:SAML:2.0:attrname-
format:unspecified">
  <saml2:AttributeValue xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:type="xsd:string">8J+YkA==</saml2:AttributeValue>
</saml2:Attribute>
```

元数据端点必须具有有效的传送层安全性

如果您在使用 HTTPS 元数据终端节点 URL 创建 SAML IdP `InvalidParameterException` 时看到，例如 “*`<metadata endpoint>`*从中检索元数据时出错”，请确保元数据端点已正确设置了 SSL，并且存在与之关联的有效的 SSL 证书。有关验证证书的更多信息，请参阅[什么是 SSL/TLS 证书？](#)。

对于 HTTP 或 HTTPS，元数据端点必须位于标准 TCP 端口上

Amazon Cognito 仅接受 HTTP URLs 的标准 TCP 端口 80 和 HTTPS 的 443 上的 SAML 提供商的元数据。作为安全最佳实践，请将 SAML 元数据托管在带有前缀的 TLS 加密的 URL 上。`https://`以*`http://www.example.com/saml2/metadata.xml`*或 URLs 的格式输入元数据*`https://www.example.com/saml2/metadata.xml`*。Amazon

Cognito 控制台 URLs 仅接受带有前`https://`缀的元数据。您也可以使用和配置 IdP 元数据。CreateIdentityProviderUpdateIdentityProvider

使用 IdP 发起的 SAML 的应用程序客户端只能使用 SAML 登录

当您激活对支持 IdP 启动登录应用程序客户端的 SAML 2.0 IdP 的支持时，您只能向该应用程序客户端添加其他 SAML IdPs 2.0。您无法将用户池中的用户目录以及所有非 SAML 外部身份提供者添加到以这种方式配置的应用程序客户端。

注销响应必须使用 POST 绑定

`/saml2/logout` 端点接受 `LogoutResponse` 作为 `HTTP POST` 请求。用户池不接受带有 `HTTP GET` 绑定的注销响应。

元数据签名证书轮换

当您提供带有 URL 的元数据时，Amazon Cognito 会将 SAML 元数据缓存长达六个小时。执行任何元数据签名证书轮换时，请将元数据源配置为发布原始证书和新证书至少六个小时。当 Amazon Cognito 从元数据 URL 刷新缓存时，它会将每个证书视为有效，并且您的 SAML IdP 可以开始使用新证书签署 SAML 断言。在此期限过后，您可以从已发布的元数据中删除原始证书。

SAML 用户名区分大小写

当联合用户尝试登录时，SAML 身份提供者（IdP）在用户的 SAML 断言中将唯一 `NameId` 传递给 Amazon Cognito。Amazon Cognito 通过其 `NameId` 声明识别 SAML 联合身份用户。无论您的用户池的区分大小写如何设置，当从 SAML IdP 返回的联合用户传递其唯一且区分大小写的 `NameId` 声明时，Amazon Cognito 都会识别该用户。如果您将 `email` 等属性映射到 `NameId`，并且您的用户更改其电子邮件地址，他们将无法登录您的应用程序。

从具有不会改变的值的 IdP 属性映射 SAML 断言中的 `NameId`。

例如，Carlos 在您的不区分大小写的用户池中具有来自 Active Directory 联合身份验证服务 (ADFS) SAML 断言的用户配置文件，该断言传递了`Carlos@example.com` 的 `NameId` 值。下次 Carlos 尝试登录时，您的 ADFS IdP 会传递`carlos@example.com` 的 `NameId` 值。由于 `NameId` 的大小写必须完全匹配，登录不成功。

如果您的用户在其 `NameID` 更改后无法登录，请从您的用户池中删除他们的用户配置文件。Amazon Cognito 将在用户下次登录时创建新的用户配置文件。

主题

- 配置第三方 SAML 身份提供者

- 在用户池中添加和力 SAML 身份提供者
- SAML 会话在 Amazon Cognito 用户池中启动
- 通过单点注销来注销 SAML 用户
- SAML 签名和加密
- SAML 身份提供者名称和标识符

配置第三方 SAML 身份提供者

如果要将 SAML 身份提供者（IdP）添加到用户池，则必须在 IdP 的管理界面中进行一些配置更新。本节介绍如何格式化您必须提供给 IdP 的值。您还可以了解如何检索用于向用户池标识 IdP 及其 SAML 声明的静态或活动 URL 元数据文档。

要配置第三方 SAML 2.0 身份提供者（IdP）解决方案以使用 Amazon Cognito 用户池的联合身份验证，您必须配置 SAML IdP 以重定向到以下断言使用者服务（ACS）URL：`https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse`。如果您的用户池有 Amazon Cognito 域，则可以在 A mazon Cognit o 控制台的用户池的域菜单中找到您的用户池域路径。

某些 SAML IdPs 要求您在表`urn:amazon:cognito:sp:us-east-1_EXAMPLE`单中`urn`提供（也称为受众 URI 或 SP 实体 ID）。您可以在 Amazon Cognito 控制台的用户池概览下找到用户池 ID。

您还必须配置 SAML IdP，以便为用户池中您指定为必需属性的任何属性提供值。通常，`email` 是用户池的必需属性，在这种情况下，SAML IdP 必须在其 SAML 断言中提供某种形式的 `email` 声明，且您必须将该声明映射到该提供者的属性。

以下第三方 SAML 2.0 IdP 解决方案的配置信息是开始使用 Amazon Cognito 用户池设置联合身份验证的一个很好的起点。有关最新信息，请直接查阅提供者的文档。

要签署 SAML 请求，必须将 IdP 配置为信任由您的用户池签名证书签署的请求。要接受加密的 SAML 响应，必须将 IdP 配置为对用户池的所有 SAML 响应进行加密。您的提供者将提供有关配置这些特征的文档。有关 Microsoft 的示例，请参阅配置 Microsoft Entra SAML 令牌加密。

> **Note**
>
> Amazon Cognito 只需要身份提供者元数据文档。您的提供商还可能为与 IAM 或 Amazon IAM Identity Center SAML 2.0 联合身份验证提供自定义配置信息。要了解如何设置 Amazon Cognito 集成，请查看检索元数据文档的一般说明并管理用户池中的其余配置。

解决方案	更多信息
Microsoft Entra ID	联邦元数据
Okta	如何下载用于 SAML 应用程序集成的 IdP 元数据和 SAML 签名证书
Auth0	将 Auth0 配置为 SAML 身份提供者
Ping 身份 (PingFederate)	从中导出 SAML 元数据 PingFederate
JumpCloud	SAML 配置备注
SecureAuth	SAML 应用程序集成

在用户池中添加和力 SAML 身份提供者

将身份提供者配置为与 Amazon Cognito 配合使用后，您可以将其添加到用户池和应用程序客户端。以下步骤演示了如何在 Amazon Cognito 用户池中创建、修改和删除 SAML 提供者。

Amazon Web Services Management Console

您可以使用 Amazon Web Services Management Console 来创建和删除 SAML 身份提供商 (IdPs)。

在创建 SAML IdP 之前，您必须具有从第三方 IdP 处获得的 SAML 元数据文档。有关如何获取或生成所需的 SAML 元数据文档的说明，请参阅配置第三方 SAML 身份提供者。

在您的用户池中配置 SAML 2.0 IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 Amazon 凭证。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “社交和外部提供商” 菜单，然后选择 “添加身份提供商”。
5. 选择一个 SAML IdP。
6. 输入提供商名称。您可以在 `identity_provider` 请求参数中将这个友好名称传递给 对端点授权。

7. 输入以逗号分隔的 Identifiers（标识符）。标识符将告知 Amazon Cognito 应该检查用户登录时输入的电子邮件地址，然后将它们引导到与其域名对应的提供商。
8. 如果您希望 Amazon Cognito 在用户注销时向您的提供商发送已签名的注销请求，请选择 Add sign-out flow（添加注销流程）。您必须将 SAML 2.0 IdP 配置为向配置托管登录时创建`https://mydomain.auth.us-east-1.amazoncognito.com/saml2/logout`的终端节点发送注销响应。此 `saml2/logout` 端点使用 POST 绑定。

   > **Note**
   >
   > 如果选择此选项，且您的 SAML IdP 需要已签署的注销请求，则还必须为您的 SAML IdP 提供来自用户池的签名证书。
   > SAML IdP 将处理已签名的注销请求并从 Amazon Cognito 会话中注销您的用户。

9. 选择 IdP 发起的 SAML 登录配置。作为一项安全最佳实践，请选择仅接受 SP 发起的 SAML 断言。如果您已准备好环境，以便安全地接受未经请求的 SAML 登录会话，请选择接受 SP 发起和 IdP 发起的 SAML 断言。有关更多信息，请参阅 SAML 会话在 Amazon Cognito 用户池中启动。
10. 选择 Metadata document source（元数据文档源）。如果您的 IdP 在公有 URL 上提供 SAML 元数据，则可以选择 Metadata document URL（元数据文档 URL），然后输入该公有 URL。否则，请选择 Upload metadata document（上载元数据文档），然后选择您之前从提供商下载的元数据文件。

    > **Note**
    >
    > 如果您的提供者具有公有端点，我们建议您输入元数据文档 URL，而不是上传文件。Amazon Cognito 会自动从元数据 URL 刷新元数据。通常，元数据刷新操作每 6 小时执行一次或在元数据过期前执行（以时间较早者为准）。

11. 在 SAML 提供者和用户池之间映射属性选项将 SAML 提供者属性映射到用户池中的用户配置文件。在属性映射中包含用户池必需属性。

    例如，当您选择 User pool attribute（用户池属性）`email` 时，按照您的 IdP 提供的 SAML 断言中显示的内容，输入 SAML 属性名称。如果 IdP 提供了示例 SAML 断言，您可以使用这些示例断言帮助您查找名称。有些 IdPs 使用简单的名称，例如`email`，而另一些则使用如下所示的名称。

    ```
    http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
    ```

12. 选择创建。

API/CLI

使用以下命令可创建和管理 SAML 身份提供商 (IdP)。

创建 IdP 并上传元数据文档

- Amazon CLI: `aws cognito-idp create-identity-provider`

  带元数据文件的示例：`aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  其中 `details.json` 包含：

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果`<SAML metadata XML>`包含该字符的任何实例"，则必须添加\为转义字符：\"。

  带元数据 URL 的示例：`aws cognito-idp create-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-type SAML --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- Amazon API: CreateIdentityProvider

为 IdP 上传新的元数据文档

- Amazon CLI: `aws cognito-idp update-identity-provider`

  带元数据文件的示例：`aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  其中 `details.json` 包含：

```
"ProviderDetails": {
      "MetadataFile": "<SAML metadata XML>",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

> **Note**
>
> 如果<SAML metadata XML>包含该字符的任何实例"，则必须添加\为转义字符：\"。

  带元数据 URL 的示例：`aws cognito-idp update-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1 --provider-details MetadataURL=https://myidp.example.com/sso/saml/metadata --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

- Amazon API: UpdateIdentityProvider

获取有关特定 IdP 的信息

- Amazon CLI: `aws cognito-idp describe-identity-provider`

  `aws cognito-idp describe-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1`

- Amazon API: DescribeIdentityProvider

列出所有相关信息 IdPs

- Amazon CLI: `aws cognito-idp list-identity-providers`

  示例：`aws cognito-idp list-identity-providers --user-pool-id us-east-1_EXAMPLE --max-results 3`

- Amazon API: ListIdentityProviders

删除 IdP

- Amazon CLI: `aws cognito-idp delete-identity-provider`

  `aws cognito-idp delete-identity-provider --user-pool-id us-east-1_EXAMPLE --provider-name=SAML_provider_1`

- Amazon API: DeleteIdentityProvider

设置 SAML IdP 以添加用户池作为信赖方

- 用户池服务提供商 URN 为：`urn:amazon:cognito:sp:us-east-1_EXAMPLE`。Amazon Cognito 要求受众限制值与 SAML 响应中的这个 URN 相匹配。将您的 IdP 配置为使用以下 POST 绑定端点 IdP-to-SP作为响应消息。

```
https://mydomain.auth.us-east-1.amazoncognito.com/saml2/idpresponse
```

- 您的 SAML IdP 必须在 SAML 断言中为用户池填入 `NameID` 和任何必需属性。`NameID` 用于在用户池中唯一地标识您的 SAML 联合用户。您的 IdP 必须以一致、区分大小写的格式传递每个用户的 SAML 名称 ID。用户名 ID 值发生任何变化都会创建一个新的用户配置文件。

向您的 SAML 2.0 IDP 提供签名证书

- 要从 Amazon Cognito 下载公钥副本，供您的 IdP 用来验证 SAML 注销请求，请选择用户池的社交和外部提供商菜单，选择您的 IdP，然后在 “查看签名证书” 下，选择下载为.crt。

您可以使用 Amazon Cognito 控制台删除在用户池中设置的任何 SAML 提供商。

删除 SAML 提供者

1. 登录 Amazon Cognito 控制台。

2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 “社交和外部提供商” 菜单。
4. 选择要删除的 SAML IdPs 旁边的单选按钮。
5. 当系统提示您 Delete identity provider（删除身份提供商）时，请输入 SAML 提供商的名称以确认删除，然后选择 Delete（删除）。

SAML 会话在 Amazon Cognito 用户池中启动

Amazon Cognito 支持服务提供商发起（SP 发起）的单点登录（SSO）和 IdP 发起的 SSO。作为一项最佳安全实践，请在用户池中实施 SP 发起的 SSO。SAML V2.0 技术概述的第 5.1.2 节描述了 SP 启动的 SSO。Amazon Cognito 是您的应用程序的身份提供商 (IdP)。该应用程序是为经过身份验证的用户检索令牌的服务提供程序 (SP)。但是，当您使用第三方 IdP 对用户进行身份验证时，Amazon Cognito 就是 SP。SAML 2.0 用户使用 SP 发起的流程进行身份验证时，他们始终必须先向 Amazon Cognito 发出请求并重定向到 IdP 进行身份验证。

对于某些企业使用案例，对内部应用程序的访问从企业 IdP 托管的控制面板上的书签开始。当用户选择书签时，IdP 会生成一个 SAML 响应并将其发送到 SP 以向应用程序验证用户身份。

您可以在用户池中配置 SAML IdP，以便支持 IdP 发起的 SSO。在支持 IdP 发起的身份验证时，因为 Amazon Cognito 不会通过 SAML 请求发起身份验证，所以 Amazon Cognito 无法验证它是否请求了收到的 SAML 响应。在 SP 发起的 SSO 中，Amazon Cognito 会设置状态参数，用以根据原始请求验证 SAML 响应。通过 SP 发起的登录，您还可以防止跨站点请求伪造（CSRF）攻击。

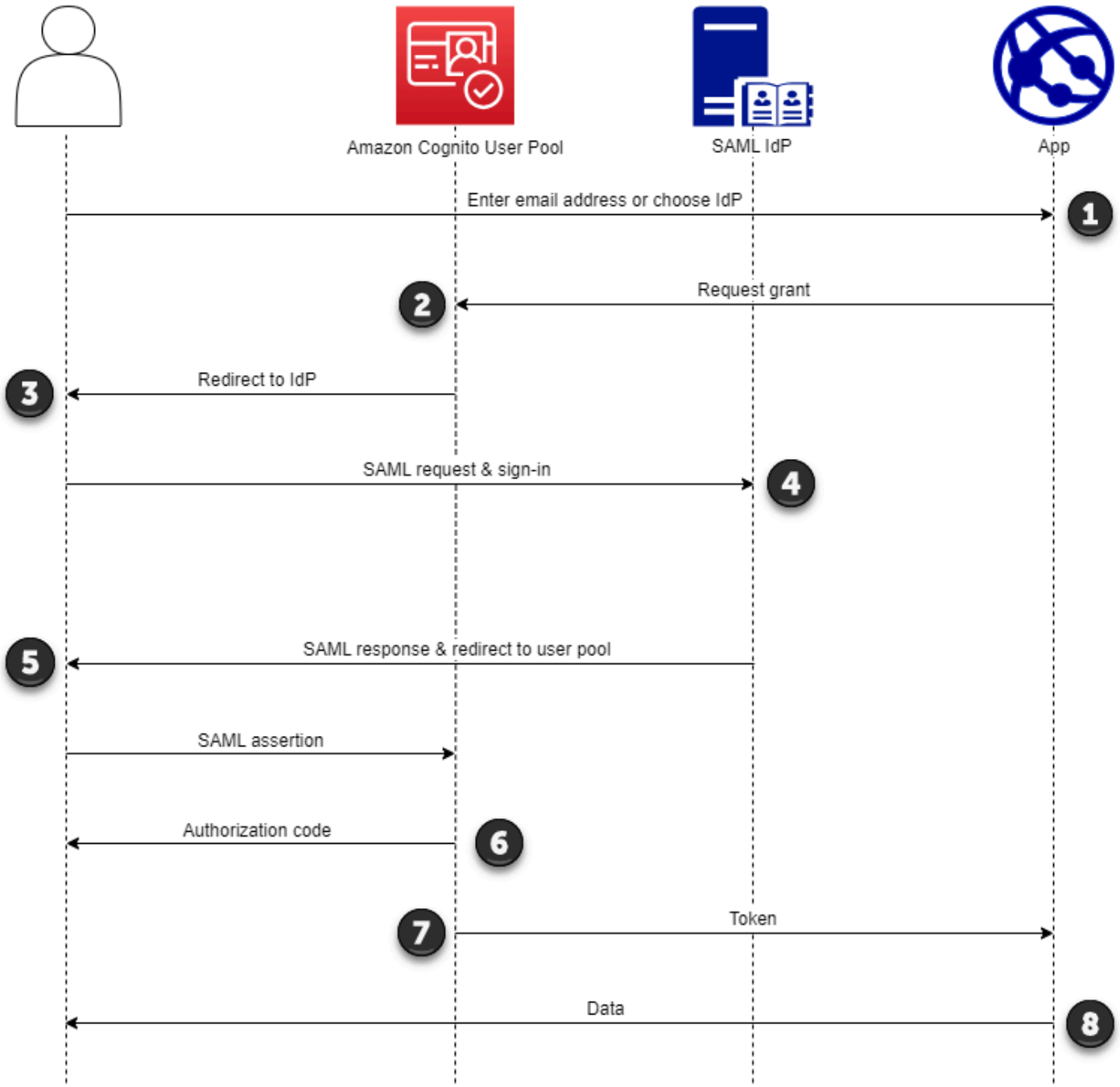
主题

- 实现 SP 启动的 SAML 登录
- 实施 IDP 发起的 SAML 登录

实现 SP 启动的 SAML 登录

最佳做法是实现 service-provider-initiated（由 SP 发起的）用户池登录。Amazon Cognito 会启动用户的会话并将他们重定向到您的 IdP。使用这种方法，您可以更大限度地控制谁提出登录请求。在某些条件下，您也可以允许 IdP 发起的登录。

以下过程显示了用户如何通过 SAML 提供者完成 SP 发起的用户池登录。



1. 您的用户在登录页面输入他们的电子邮件地址。要确定您的用户是否重定向到其 IdP，您可以在自定义应用程序中收集他们的电子邮件地址，或者在 Web 视图中调用托管登录。

   您可以将托管登录页面配置为显示电子邮件地址列表 IdPs或提示输入电子邮件地址，然后将其与 SAML IdP 的标识符进行匹配。要提示输入电子邮件地址，请编辑您的托管登录品牌样式，然后在 Foundati on 中找到 “身份验证行为”，然后在 “提供商显示” 下，将显示样式设置为域搜索输入。

2. 您的应用程序调用您的用户池重定向端点，并请求使用与应用程序对应的客户端 ID 以及与用户对应的 IdP ID 进行会话。
3. Amazon Cognito 使用（可选择在 `AuthnRequest` 元素中签署的）SAML 请求将您的用户重定向到 IdP。
4. IdP 以交互方式或通过浏览器 Cookie 中记住的会话对用户进行身份验证。
5. IdP 使用其 POST 有效载荷中可选的加密 SAML 断言将用户重定向到用户池 SAML 响应端点。

> **Note**
>
> Amazon Cognito 会取消在 5 分钟内未收到回复的会话，并将用户重定向到托管登录。当您的用户遇到此结果时，他们会收到一条 `Something went wrong` 错误消息。

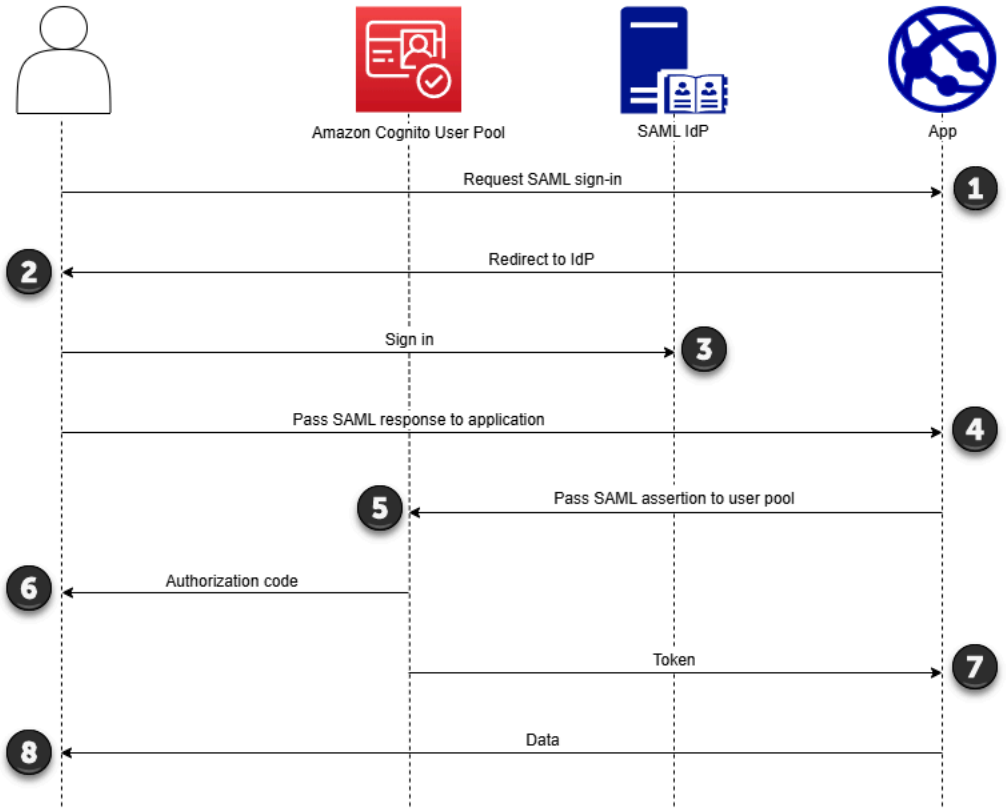
6. 在验证 SAML 断言并从响应中的声明映射用户属性后，Amazon Cognito 在用户池中内部创建或更新用户的配置文件。通常，您的用户池会向用户的浏览器会话返回授权码。
7. 您的用户向您的应用程序出示他们的授权码，您的应用程序会将该代码兑换 JSON 网络令牌 (JWTs)。
8. 应用程序接受并处理用户的 ID 令牌作为身份验证，使用其访问令牌生成对资源的授权请求，并存储他们的刷新令牌。

当用户进行身份验证并接收授权码授予时，用户池会返回 ID 令牌、访问令牌和刷新令牌。ID 令牌是基于 OIDC 的身份管理的身份验证对象。访问令牌是一个范围为 OAuth 2.0 的授权对象。刷新令牌是在用户当前令牌到期时生成新 ID 令牌和访问令牌的对象。您可以在用户池应用程序客户端中配置用户令牌的持续时间。

您还可以选择刷新令牌的持续时间。用户的刷新令牌到期后，他们必须重新登录。如果他们通过 SAML IdP 进行身份验证，则用户的会话持续时间由其令牌的到期时间，而不是他们与 IdP 的会话到期时间来设置。您的应用程序必须存储每位用户的刷新令牌，并在刷新令牌到期时更新他们的会话。托管登录在有效期为 1 小时的浏览器 Cookie 中维护用户会话。

实施 IDP 发起的 SAML 登录

在为 IdP 发起的 SAML 2.0 登录配置身份提供者时，您可以向用户池域中的 `saml2/idpresponse` 端点提供 SAML 断言，而无需在 对端点授权 启动会话。具有此配置的用户池接受从用户池外部身份提供者由 IdP 发起的 SAML 断言，前提是所请求的应用程序客户端支持这种断言。



1. 用户请求使用您的应用程序进行 SAML 登录。
2. 您的应用程序调用浏览器或将用户重定向到其 SAML 提供商的登录页面。
3. IdP 以交互方式或通过浏览器 Cookie 中记住的会话对用户进行身份验证。
4. IdP 在其 POST 正文中使用 SAML 断言或响应将您的用户重定向到您的应用程序。
5. 您的应用程序将 SAML 断言添加到您的用户池`saml2/idpresponse`终端节点请求的 POST 正文中。
6. Amazon Cognito 会向您的用户发出授权码。
7. 您的用户向您的应用程序出示他们的授权码，您的应用程序会将该代码兑换 JSON 网络令牌 (JWTs)。
8. 您的应用程序接受并处理用户的 ID 令牌作为身份验证，使用其访问令牌生成对资源的授权请求，并存储他们的刷新令牌。

以下步骤描述了配置 IdP 发起的 SAML 2.0 提供者并使用它来登录的整个过程。

1. 创建或指定用户池和应用程序客户端。
2. 在您的用户池中创建 SAML 2.0 IdP。

3. 将您的 IdP 配置为支持 IdP 启动。IdP 发起的 SAML 引入了一些安全性考量，而其他 SSO 提供者没有涉及这些安全性考量。因此，您无法将非 SAML IdPs（包括用户池本身）添加到任何使用 SAML 提供商并通过 IDP 启动登录的应用程序客户端。
4. 将 IdP 发起的 SAML 提供者与用户池中的应用程序客户端关联。
5. 将您的用户引导至 SAML IdP 的登录页面并检索 SAML 断言。
6. 使用 SAML 断言将用户引导至用户池 `saml2/idpresponse` 端点。
7. 接收 JSON 网络令牌 (JWTs)。

要在用户池中接受未经请求的 SAML 断言，必须考虑其对应用程序安全性的影响。当您接受 IdP 发起的请求时，可能会出现请求欺骗和 CSRF 尝试情况。虽然用户池无法验证 IdP 发起的登录会话，但 Amazon Cognito 会验证请求参数和 SAML 断言。

此外，SAML 声明不得包含 `InResponseTo` 声明，并且必须在前 6 分钟内发出。

您必须使用 IdP 发起的 SAML 向您的 `/saml2/idpresponse` 提交请求。对于 SP 发起的托管登录授权请求，您必须提供参数，这些参数将您请求的应用程序客户端、范围、重定向 URI 和其他详细信息标识为请求中的`HTTP GET`查询字符串参数。但是，对于 IdP 发起的 SAML 断言，必须将请求的详细信息的格式设置为 `HTTP POST` 请求正文中的 `RelayState` 参数。请求正文还必须包含您的 SAML 断言，作为 `SAMLResponse` 参数。

以下是 IDP 发起的 SAML 提供商的请求和响应示例。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone

HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

Amazon Web Services Management Console

为 IdP 发起的 SAML 配置 IdP

1. 创建用户池、应用程序客户端和 SAML 身份提供者。
2. 取消所有社交和 OIDC 身份提供者与应用程序客户端的关联（如果已关联）。
3. 导航到用户池的 “社交和外部提供商” 菜单。
4. 编辑或添加 SAML 提供商。
5. 在 IdP 发起的 SAML 登录下，选择接受 SP 发起和 IdP 发起的 SAML 断言。
6. 选择保存更改。

API/CLI

为 IdP 发起的 SAML 配置 IdP

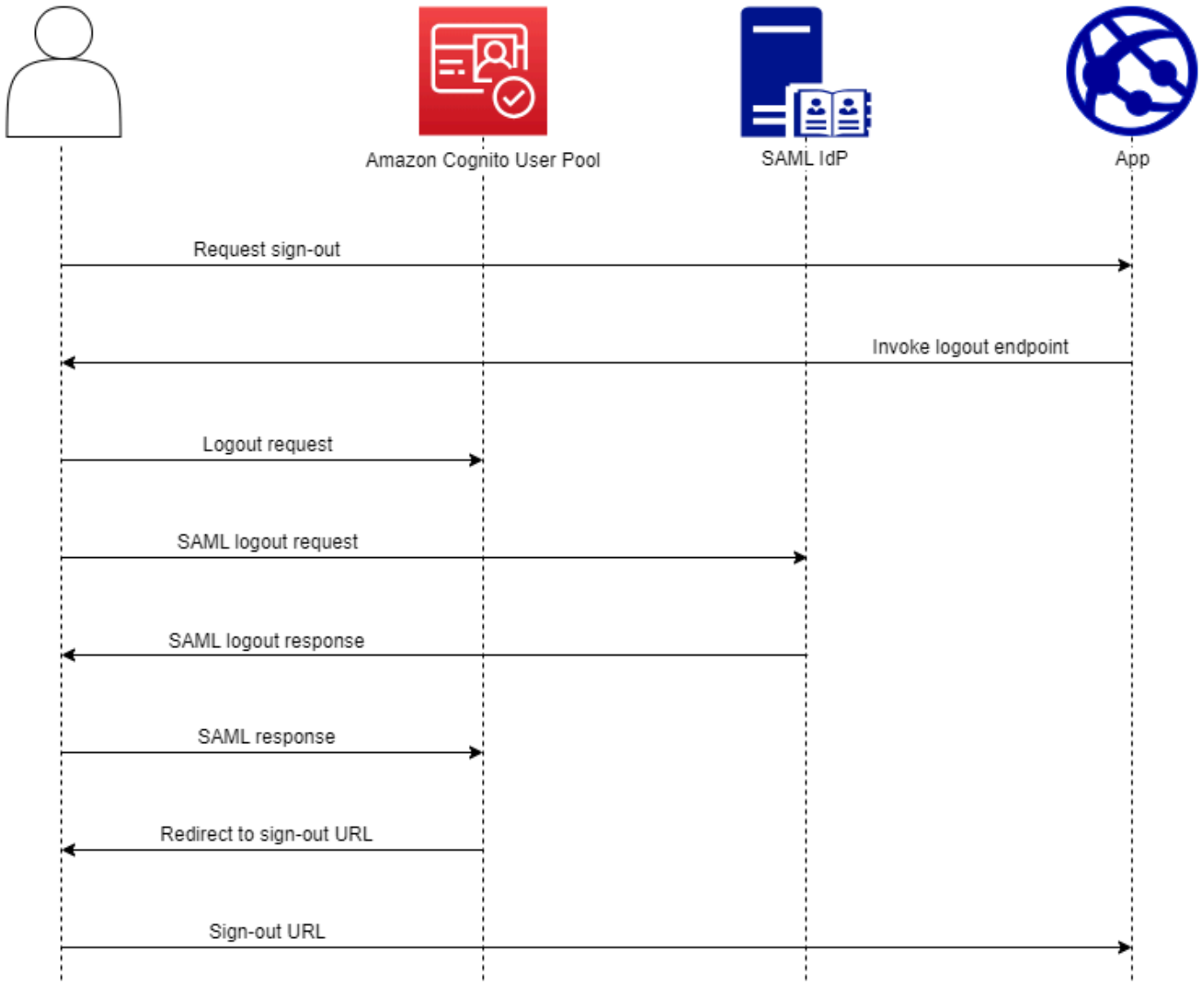
使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求中的`IDPInit`参数配置 IDP 启动的 SAML。以下是支持 IdP 发起的 SAML 的 IdP 的示例 `ProviderDetails`。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

通过单点注销来注销 SAML 用户

Amazon Cognito 支持 SAML 2.0 单点注销 (SLO)。 借助 SLO，当用户从您的用户池中注销时，您的应用程序可以从其 SAML 身份提供商 (IdPs) 中注销他们。这样，当用户想要再次登录应用程序时，他们必须使用其 SAML IdP 进行身份验证。如果不这样做，他们可能会因为在浏览器中有 IdP 或用户池 Cookie，所以无需提供凭证即可进入您的应用程序。

当您将 SAML IdP 配置为支持注销流程时，Amazon Cognito 会通过已签名的 SAML 注销请求将您的用户重定向到您的 IdP。Amazon Cognito 根据您的 IdP 元数据中的 `SingleLogoutService` URL 确定重定向位置。Amazon Cognito 使用您的用户池签名证书签署注销请求。



当您将具有 SAML 会话的用户定向到您的用户池 `/logout` 端点时，Amazon Cognito 会将您的 SAML 用户通过以下请求重定向到 IdP 元数据中指定的 SLO 端点。

```
https://[SingleLogoutService endpoint]?
SAMLRequest=[encoded SAML request]&
RelayState=[RelayState]&
SigAlg=http://www.w3.org/2001/04/xmldsig-more#rsa-sha256&
Signature=[User pool RSA signature]
```

然后，您的用户使用来自其 IdP 的 `LogoutResponse` 返回到您的 `saml2/logout` 端点。您的 IdP 必须在 `HTTP POST` 请求中发送 `LogoutResponse`。然后，Amazon Cognito 会根据他们最初的注销请求将他们重定向到重定向目的地。

您的 SAML 提供者可能会发送包含多个 `AuthnStatement` 的 `LogoutResponse`。此类响应中第一个 `AuthnStatement` 中的 `sessionIndex` 必须与最初对用户进行身份验证的 SAML 响应中的 `sessionIndex` 匹配。如果 `sessionIndex` 在任何其他 `AuthnStatement` 中，则 Amazon Cognito 将无法识别会话，且您的用户不会注销。

Amazon Web Services Management Console

配置 SAML 注销

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑 SAML 身份提供者时，在身份提供商信息下，选中名称为添加注销流程的复选框。
3. 从用户池的社交和外部提供商菜单中，选择您的 IdP 并找到签名证书。
4. 选择以 .crt 格式下载证书。
5. 将您的 SAML 提供者配置为支持 SAML 单点注销和请求签名，然后上传用户池签名证书。您的 IdP 必须重定向到用户池域中的 `/saml2/logout`。

API/CLI

配置 SAML 注销

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的`IDPSignout`参数配置单次注销。以下是支持 SAML 单点注销的 IdP 的示例 `ProviderDetails`。

```
"ProviderDetails": {
      "MetadataURL" : "https://myidp.example.com/saml/metadata",
      "IDPSignout" : "true",,
      "RequestSigningAlgorithm" : "rsa-sha256",
      "EncryptedResponses" : "true",
      "IDPInit" : "true"
}
```

SAML 签名和加密

SAML 2.0 登录围绕着应用程序的用户作为其身份验证流程中的请求和响应的持有者来进行。您可能需要确保用户不会在传输过程中读取或修改这些 SAML 文档。为此，请将 SAML 签名和加密添加到用户池中的 SAML 身份提供商 (IdPs)。借助 SAML 签名，您的用户池可向 SAML 登录和注销请求添加签名。使用您的用户池公钥，IdP 可以验证它接收的是未经修改的 SAML 请求。然后，当您的 IdP 响应并将 SAML 断言传递给用户的浏览器会话时，IdP 可以加密该响应，这样用户就无法查看自己的属性和权限。

使用 SAML 签名和加密，用户池 SAML 操作期间的所有加密操作都必须使用 user-pool-provided Amazon Cognito 生成的密钥生成签名和密文。目前，无法将用户池配置为使用外部密钥签署请求或接受加密断言。

> **Note**
>
> 用户池证书有效期为 10 年。Amazon Cognito 每年都会为用户池生成一次新的签名和加密证书。请求签名证书时，Amazon Cognito 会返回最新的证书，并使用最新的签名证书来签署请求。IdP 可以使用任何未过期的用户池加密证书来加密 SAML 断言。之前的证书在整个有效期内继续有效，并且不同证书之间的公钥不会发生变化。作为一项最佳实践，请每年更新提供者配置中的证书。

主题

- 接受来自 IdP 的加密 SAML 响应
- 签署 SAML 请求

接受来自 IdP 的加密 SAML 响应

当用户登录和注销时，Amazon Cognito 和您的 IdP 可以在 SAML 响应中保持机密性。Amazon Cognito 会向您在用户池中配置的每个外部 SAML 提供者分配一个公有-私有 RSA 密钥对和一个证书。为用户池 SAML 提供者启用响应加密时，必须将证书上传到支持加密 SAML 响应的 IdP。在您的 IdP 开始使用提供的密钥加密所有 SAML 断言之前，您的用户池与 SAML IdP 的连接无法正常工作。

下文概述了加密 SAML 登录的流程。

1. 您的用户开始登录并选择他们的 SAML IdP。

2. 您的用户池 对端点授权 通过 SAML 登录请求将您的用户重定向到他们的 SAML IdP。您的用户池可以选择在此请求中附上签名，从而让 IdP 可以进行完整性验证。当您想要签署 SAML 请求时，必须将您的 IdP 配置为接受您的用户池使用签名证书中的公钥签署的请求。
3. SAML IdP 让您的用户登录并生成 SAML 响应。IdP 使用公钥对响应进行加密，并将用户重定向到用户池 `/saml2/idpresponse` 端点。IdP 必须按照 SAML 2.0 规范的定义对响应进行加密。有关更多信息，请参阅 OASIS 安全断言标记语言（SAML）V2.0 的断言和协议中的 `Element <EncryptedAssertion>`。
4. 用户池使用私钥解密 SAML 响应中的加密文字并让您的用户登录。

> **Important**
>
> 当您为用户池中的 SAML IdP 启用响应加密时，您的 IdP 必须使用该提供者特有的公钥对所有响应进行加密。Amazon Cognito 不接受来自您配置为支持加密的 SAML 外部 IdP 的未加密 SAML 响应。

用户池中的任何外部 SAML IdP 都可以支持响应加密，并且每个 IdP 都会收到自己的密钥对。

Amazon Web Services Management Console

配置 SAML 响应加密

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑 SAML 身份提供者时，在对请求进行签名并加密响应下，选中名称为需要此提供商的加密 SAML 断言的复选框。
3. 从用户池的社交和外部提供商菜单中，选择您的 SAML IdP，然后选择查看加密证书。
4. 选择以 .crt 格式下载证书，然后将下载的文件提供给您的 SAML IdP。将您的 SAML IdP 配置为使用证书中的密钥加密 SAML 响应。

API/CLI

配置 SAML 响应加密

使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的`EncryptedResponses`参数配置响应加密。以下是支持请求签名的 IdP 的示例 `ProviderDetails`。

```
"ProviderDetails": {
```

```
        "MetadataURL" : "https://myidp.example.com/saml/metadata",
        "IDPSignout" : "true",
        "RequestSigningAlgorithm" : "rsa-sha256",
        "EncryptedResponses" : "true",
        "IDPInit" : "true"
}
```

要从您的用户池中获取加密证书，请发出 DescribeIdentityProviderAPI 请求并在响应参数`ActiveEncryptionCertificate`中检索的值`ProviderDetails`。保存此证书，并将其作为加密证书提供给您的 IdP，用于接收来自用户池的登录请求。

签署 SAML 请求

能够向您的 IdP 证明 SAML 2.0 请求的完整性是 Amazon Cognito SP 发起的 SAML 登录的一项安全优势。每个拥有域的用户池都会收到一个用户池 X.509 签名证书。使用此证书中的公钥，用户池将加密签名应用于在用户选择 SAML IdP 时用户池生成的注销请求。您可以选择将应用程序客户端配置为签署 SAML 登录请求。当您签署 SAML 请求时，您的 IdP 会核实请求的 XML 元数据中的签名是否与您提供的用户池证书中的公钥相匹配。

Amazon Web Services Management Console

配置 SAML 请求签名

1. 创建用户池、应用程序客户端和 SAML IdP。
2. 创建或编辑 SAML 身份提供者时，在对请求进行签名并加密响应下，选中名称为对此提供商的 SAML 请求进行签名的复选框。
3. 从用户池的 “社交和外部提供商” 菜单中，选择 “查看签名证书”。
4. 选择以 .crt 格式下载证书，然后将下载的文件提供给您的 SAML IdP。配置 SAML IdP 以验证传入 SAML 请求的签名。

API/CLI

配置 SAML 请求签名

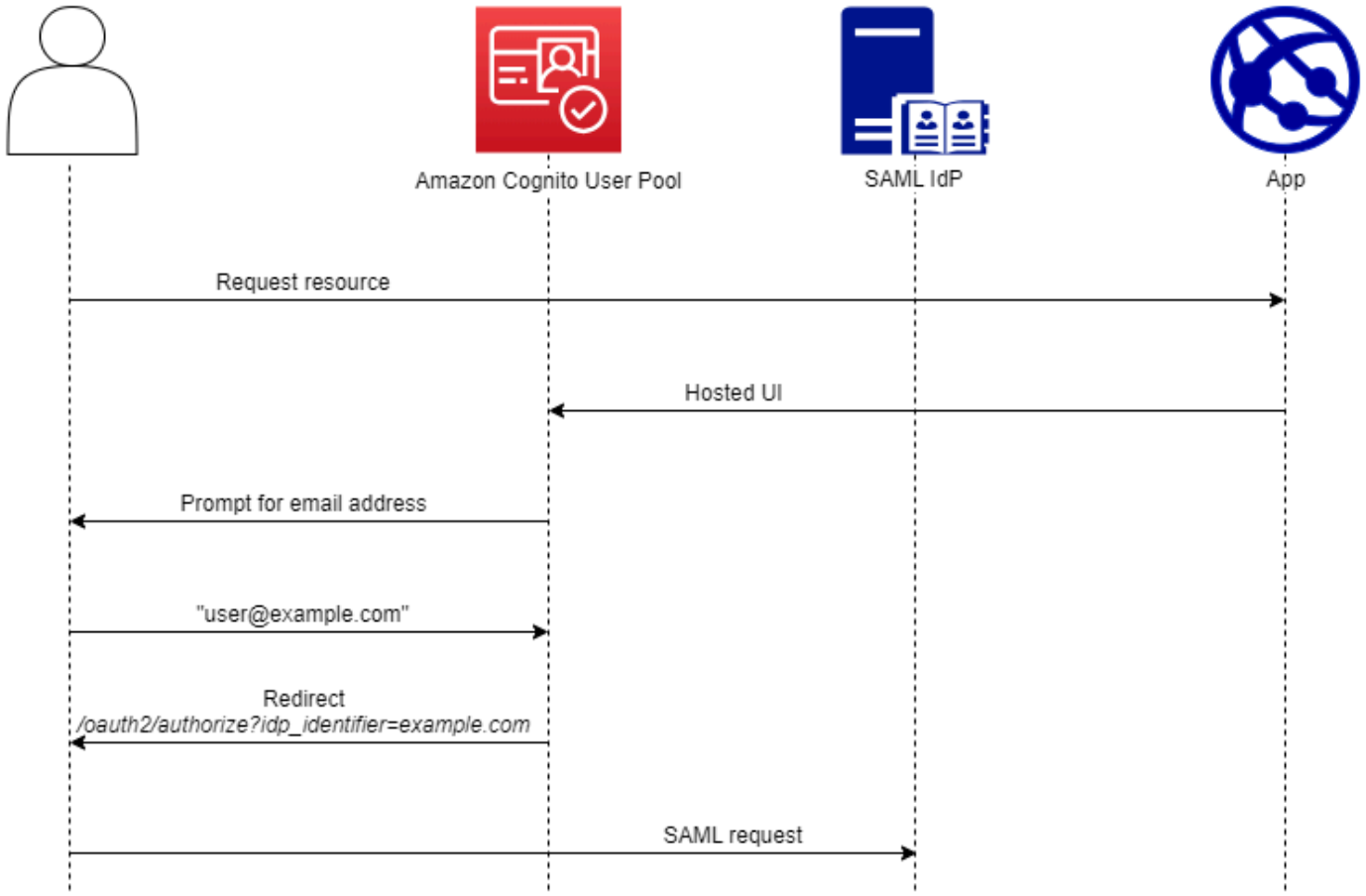
使用CreateIdentityProvider或 UpdateIdentityProviderAPI 请求的`RequestSigningAlgorithm`参数配置请求签名。以下是支持请求签名的 IdP 的示例 `ProviderDetails`。

```
"ProviderDetails": {
```

```
        "MetadataURL" : "https://myidp.example.com/saml/metadata",
        "IDPSignout" : "true",
        "RequestSigningAlgorithm" : "rsa-sha256",
        "EncryptedResponses" : "true",
        "IDPInit" : "true"
}
```

SAML 身份提供者名称和标识符

当您命名您的 SAML 身份提供商 (IdPs) 并分配 IdP 标识符时，您可以自动将 SP 发起的登录和注销请求流向该提供商。有关提供商名称的字符串约束的信息，请参阅的`ProviderName`属性CreateIdentityProvider。



您还可以为 SAML 提供商选择最多 50 个标识符。标识符是用户池中 IdP 的友好名称，并且在用户池中必须是唯一的。如果您的 SAML 标识符与用户的电子邮件域名相匹配，则托管登录会请求每个用户的电子邮件地址，评估其电子邮件地址中的域，然后将他们重定向到与其域名对应的 IdP。由于同一个组织可以拥有多个域，因此单个 IdP 可以有多个标识符。

无论您是使用还是不使用电子邮件域标识符，您都可以在多租户应用程序中使用标识符将用户重定向到正确的 IdP。如果您想完全绕过托管登录，则可以自定义向用户提供的链接，以便他们通过对端点授权直接重定向到其 IdP。要使用标识符使您的用户登录并重定向到他们的 IdP，请在其初始授权请求的请求参数中包括 `idp_identifier=myidp.example.com` 格式的标识符。

将用户传递到 IdP 的另一种方法是使用以下 URL 格式的 IdP 名称填充参数 `identity_provider`。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
identity_provider=MySAMLIdP&
client_id=1example23456789&
redirect_uri=https://www.example.com
```

用户使用您的 SAML IdP 登录后，您的 IdP 会将他们重定向到您的 `/saml2/idpresponse` 端点，并在 HTTP POST 正文中包括一个 SAML 响应。Amazon Cognito 会处理 SAML 断言，如果响应中的声明符合预期，则会重定向到您的应用程序客户端回调 URL。在用户以这种方式完成身份验证后，他们只与您的 IdP 和您的应用程序的网页进行交互。

使用域格式的 IdP 标识符，托管登录会在登录时请求电子邮件地址，然后，当电子邮件域与 IdP 标识符匹配时，会将用户重定向到其 IdP 的登录页面。例如，您构建一个需要两家不同公司的员工登录的应用程序。第一家公司 AnyCompany A 拥有`exampleA.com`和`exampleA.co.uk`。第二家公司 AnyCompany B 拥有`exampleB.com`。在本示例中，您设置了两个 IdPs，每家公司一个，如下所示：

- 对于 IdP A，您定义标识符 `exampleA.com` 和 `exampleA.co.uk`。
- 对于 IdP B，您定义标识符 `exampleB.com`。

在您的应用程序中，为您的应用程序客户端调用托管登录，以提示每位用户输入其电子邮件地址。Amazon Cognito 根据电子邮件地址得出域，将该域与具有域标识符的 IdP 关联起来，然后通过向包含 `idp_identifier` 请求参数的 对端点授权 发出请求，将您的用户重定向到正确的 IdP。例如，如果用户输入 `bob@exampleA.co.uk`，则他们与之交互的下一个页面是位于 `https://auth.exampleA.co.uk/sso/saml` 的 IdP 登录页面。

您也可以独立实施相同的逻辑。在您的应用程序中，您可以构建一个自定义表单，用于收集用户输入并根据自己的逻辑将其与正确的 IdP 关联起来。您可以为每个应用程序租户生成自定义门户，其中每个租户都链接到请求参数中包含租户标识符的授权端点。

要在托管登录中收集电子邮件地址并解析域，请为分配给应用程序客户端的每个 SAML IdP 分配至少一个标识符。默认情况下，托管登录屏幕会针对您分配给应用程序客户端 IdPs的每个登录显示一个按钮。但是，如果您成功分配了标识符，则您的经典托管用户界面登录页面将如下图所示。

> **Note**
>
> 在经典托管用户界面中，当您为应用程序客户端分配标识符时，应用程序客户端的登录页面会自动提示您输入电子邮件地址。 IdPs在托管登录体验中，您必须在品牌编辑器中启用此行为。在 “身份验证行为设置” 类别中，在 “提供者显示” 标题下选择 “域搜索输入”。

托管登录中的域名解析要求您使用域作为 IdP 标识符。如果您为应用程序客户端的每个 SAML IdPs 分配了任何类型的标识符，则该应用程序的托管登录将不再显示 IDP 选择按钮。当您打算使用电子邮件解析或自定义逻辑生成重定向时，请为 SAML 添加 IdP 标识符。如果您想生成静默重定向，同时希望托管登录页面显示列表，请不要分配标识符 IdPs，也不要在授权`identity_provider`请求中使用请求参数。

- 如果您只为应用程序客户端分配一个 SAML IdP，则托管登录登录页面会显示一个使用该 IdP 登录的按钮。
- 如果您为应用程序客户端激活的每个 SAML IdP 分配一个标识符，则托管登录登录页面中会出现用户输入电子邮件地址的提示。
- 如果您有多个 IdPs ，但没有为所有用户分配标识符，则托管登录页面会显示一个按钮，用于使用每个分配的 IdP 进行登录。
- 如果您为自己分配了标识符， IdPs 并且希望托管登录页面显示精选的 IdP 按钮，请向您的应用程序客户端添加一个没有标识符的新 IdP，或者创建一个新的应用程序客户端。您也可以删除现有 IdP，然后在没有标识符的情况下重新添加它。如果您创建新的 IdP，则您的 SAML 用户将创建新的用户配置文件。活跃用户的重复计数可能会在您更改 IdP 配置的当月对账单产生影响。

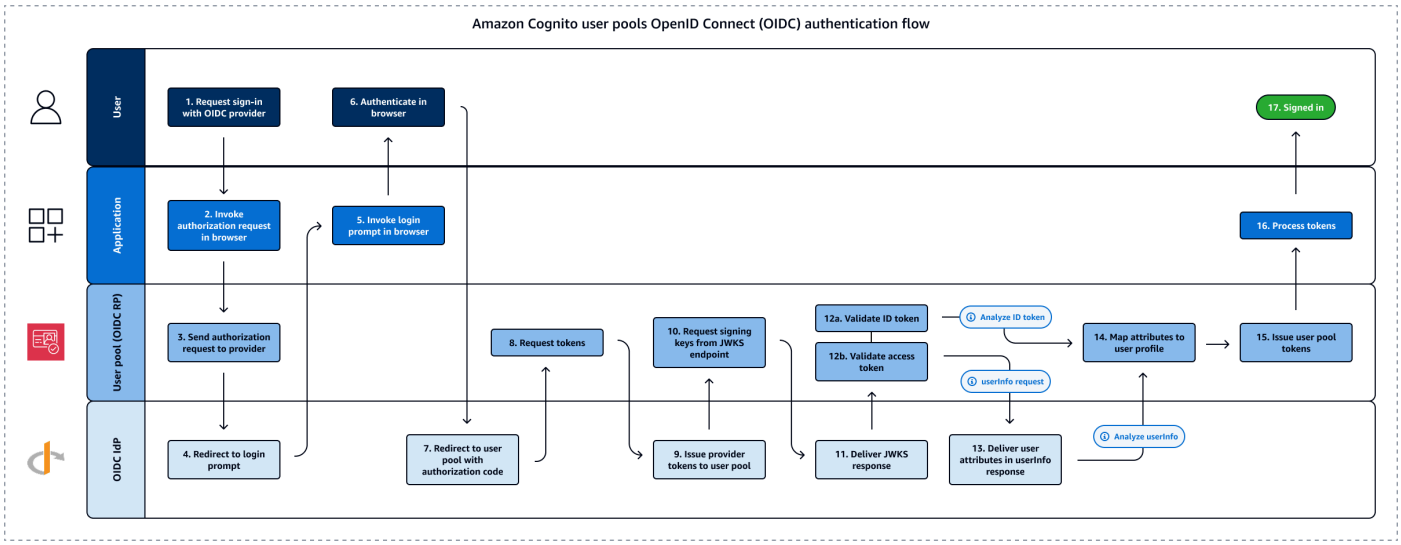
有关 IdP 设置的更多信息，请参阅为用户池配置身份提供商。

将 OIDC 身份提供者与用户池配合使用

用户可以使用 OpenID Connect (OIDC) 身份提供商 () 提供的现有账户登录您的应用程序。IdPs借助 OIDC 提供者，独立单点登录系统的用户可以提供现有凭证，同时您的应用程序以用户池共享格式接收 OIDC 令牌。要配置 OIDC IdP，请将您的 IdP 设置为将您的用户池作为 RP 处理，并将您的应用程序配置为以 IdP 的身份处理您的用户池。Amazon Cognito 是多个 OIDC IdPs 和您的应用程序之间的中间步骤。您的用户池将属性映射规则应用于您的提供商直接传递给您的用户池的 ID 和访问令牌中的声明。然后，Amazon Cognito 会根据映射的用户属性以及您使用 Lambda 触发器对身份验证流程所做的任何其他调整来发布新令牌。

使用 OIDC IdP 登录的用户无需提供新的凭证或信息即可访问您的用户池应用程序。您的应用程序可以静默地将他们重定向到其 IdP 进行登录，在后台使用用户池作为工具，用于标准化应用程序的令牌格式。要了解有关 IdP 重定向的更多信息，请参阅对端点授权。

与其他第三方身份提供者一样，您必须向 OIDC 提供者注册您的应用程序，并获取有关要连接到用户池的 IdP 应用程序的信息。用户池 OIDC IdP 需要客户端 ID、客户端密钥、您要请求的范围以及有关提供者服务端点的信息。您的用户池可以从发现端点发现提供者 OIDC 端点，也可以手动输入它们。您还必须检查提供者 ID 令牌，并在 IdP 和用户池中的属性之间创建属性映射。



有关此身份验证流程OIDC 用户池 IdP 身份验证流程的更多详细信息，请参阅。

> **Note**
>
> 通过第三方（联合身份验证）进行登录可在 Amazon Cognito 用户池中实现。此特征不依赖于通过 Amazon Cognito 身份池实现的 OIDC 联合身份验证。

您可以通过 Amazon Web Services Management Console、或使用用户池 API 方法将 OIDC IdP 添加到您的用户池中。 Amazon CLICreateIdentityProvider

主题

- 先决条件
- 向 OIDC IdP 注册应用程序
- 将 OIDC IdP 添加到用户池
- 测试 OIDC IdP 配置

- OIDC 用户池 IdP 身份验证流程

先决条件

在开始之前，您需要：

- 具有应用程序客户端和用户池域的用户池。有关更多信息，请参阅创建用户池。
- 具有以下配置的 OIDC IdP：
  - 支持 `client_secret_post` 客户端身份验证。Amazon Cognito 不在 IdP 的 OIDC 发现端点上检查 `token_endpoint_auth_methods_supported` 声明。Amazon Cognito 不支持 `client_secret_basic` 客户端身份验证。有关客户端验证的更多信息，请参阅 OpenID Connect 文档中的客户端身份验证。
  - 仅对 OIDC 端点使用 HTTPS，例如 `openid_configuration`、`userInfo` 和 `jwks_uri`。
  - 仅为 OIDC 端点使用 TCP 端口 80 和 443。
  - 只能使用 HMAC-SHA、ECDSA 或 RSA 算法对 ID 令牌进行签名。
  - 在密钥的 `jwks_uri` 处发布密钥 ID `kid` 声明，并在其令牌中包含 `kid` 声明。
  - 提供具有有效根 CA 信任链的未过期公钥。

向 OIDC IdP 注册应用程序

在将 OIDC IdP 添加到用户池配置并将其分配给应用程序客户端之前，您需要在 IdP 中设置一个 OIDC 客户端应用程序。您的用户池是用于管理您的 IdP 身份验证的中继方应用程序。

向 OIDC IdP 注册

1. 使用 OIDC IdP 创建开发人员账户。

   与 OIDC 的链接 IdPs

OIDC IdP	如何安装	OIDC 发现 URL
Salesforce	作为 OpenID Connect 身份提供商的 Salesforce	`https://MyDomainName.my.salesforce.com/.well-known/openid-configuration`

OIDC IdP	如何安装	OIDC 发现 URL
OneLogin	Connect 启用了 OIDC 的应用程序	`https://your-domain.onelogin.com/oidc/2/.well-known/openid-configuration`
JumpCloud	通过 OIDC 进行单点登录	`https://oauth.id.jumpcloud.com/.well-known/openid-configuration`
Okta	安装 Okta 身份提供商	`https://Your Okta subdomain.okta.com/.well-known/openid-configuration`
Microsoft Entra ID	微软身份平台上的 OpenID Connect	`https://login.microsoftonline.com/{tenant}/v2.0` 的值`tenant`可以包括租户 ID `common`、`organizations` 、或`consumers` 。

2. 向 OIDC IdP 注册具有 `/oauth2/idpresponse` 端点的用户池域 URL。这将确保 OIDC IdP 之后在对用户进行身份验证时通过 Amazon Cognito 接受此 URL。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
```

3. 选择您希望用户目录与用户池共享的范围。OIDC 需要使用作用域 openid IdPs 才能提供任何用户信息。需要`email`范围才能授予对`email`和`email_verified`主张的访问权限。OIDC 规范中的其他作用域`profile`适用于所有用户属性以及和`phone`。`phone_number` `phone_number_verified`
4. OIDC IdP 为您提供客户端 ID 和客户端密钥。记下这些值并将其添加到稍后添加到用户池中的 OIDC IdP 的配置中。

示例：使用 Salesforce 作为用户池的 OIDC IdP

当您要在与 OIDC 兼容的 IdP（如 Salesforce）和您的用户池之间建立信任时，请使用 OIDC IdP。

1. 在 Salesforce 开发人员网站上创建账户。
2. 通过在上一步中设置的开发人员账户登录。
3. 请在 Salesforce 页面上，执行以下操作之一：
   - 如果您使用的是 Lightning Experience，请选择设置齿轮图标，然后选择 Setup Home（设置主页）。
   - 如果您使用的是 Salesforce Classic 并且在用户界面标题中看到 Setup（设置），请选择它。
   - 如果您使用的是 Salesforce Classic 但没有在用户界面标题中看到 Setup（设置），请从顶部导航栏中选择您的姓名，然后从下拉列表中选择 Setup（设置）。
4. 在左侧导航栏上，选择 Company Settings（公司设置）。
5. 在导航栏上，选择 Domain（域），输入一个域，然后选择 Create（创建）。
6. 在左侧导航栏上，选择在 Platform Tools（平台工具）下的 Apps（应用程序）。
7. 选择 App Manager（应用程序管理器）。
8. a. 选择 New connected app（新连接的应用程序）。

   b. 完成必填句段。

      在 Start URL（启动 URL）下，在 `/authorize` 终端节点处输入使用您的 Salesforce IdP 登录的用户池域的 URL。当您的用户访问您连接的应用程序时，Salesforce 会将他们定向到此 URL 以完成登录。然后 Salesforce 将用户重定向到与应用程序客户端关联的回调 URL。

      ```
      https://mydomain.auth.us-east-1.amazoncognito.com/authorize?
      response_type=code&client_id=<your_client_id>&redirect_uri=https://
      www.example.com&identity_provider=CorpSalesforce
      ```

   c. 启用OAuth 设置并在回调 URL 中输入您的用户池域的**`/oauth2/idpresponse`**终端节点 URL。这是 Salesforce 发布授权码的网址，Amazon Cognito 用该代码兑换令牌。 OAuth

      ```
      https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/idpresponse
      ```

9. 选择 scopes（范围）。您必须包含范围 openid。要授予对 email 和 email_verified 声明的访问权限，请添加 email（电子邮件）范围。通过空格分隔范围。
10. 选择 Create（创建）。

在 Salesforce 中，客户端 ID 称为 Consumer Key（使用者密钥），客户端密钥为 Consumer Secret（使用者私有密钥）。记下您的客户端 ID 和客户端密钥。您将在下一节中使用它们。

将 OIDC IdP 添加到用户池

设置 IdP 后，您可以将用户池配置为使用 OIDC IdP 处理身份验证请求。

Amazon Cognito console

在控制台中添加 OIDC IdP

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 Amazon 凭证。
2. 从导航菜单中选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “社交和外部提供商” 菜单，然后选择 “添加身份提供商”。
5. 选择一个 OpenID Connect IdP。
6. 输入唯一的提供商名称。
7. 输入 IdP 客户端 ID。这是您在 OIDC IdP 中构建的应用程序客户端的 ID。您提供的客户端 ID 必须是您配置的回传 URL 的 OIDC 提供商。`https://[your user pool domain]/oauth2/idpresponse`
8. 输入 IdP 客户机密钥。这必须是上一步中同一应用程序客户端的客户端密钥。
9. 为该提供商输入 Authorized scopes（授权范围）。范围定义了应用程序将向您的提供商请求的用户属性组（例如 `name` 和 `email`）。按照 OAuth2.0 规范，作用域必须用空格分隔。

   您的 IdP 可能会提示用户在登录时同意向您的应用程序提供这些属性。
10. 选择属性请求方法。 IdPs可能需要将其`userInfo`终端节点的请求格式设置为`GET`或`POST`。例如，Amazon Cognito `userInfo` 终端节点需要`HTTP GET`请求。
11. 选择一种设置方法，让您的用户池确定您的 IdP 上关键 OIDC 联邦端点的路径。通常，在颁发者基本 URL 上 IdPs 托管`/well-known/openid-configuration`终端节点。如果您的提供商遇到这种情况，则自动填写发行商网址选项会提示您输入该基本网址，尝试从那里访问该`/well-known/openid-configuration`路径，然后读取那里列出的端点。您可能有非典型的终端节点路径，或者希望通过备用代理将请求传递到一个或多个终端节点。在这种情况下，请选择手动输入并指定`authorization`、`tokenuserInfo`、和`jwks_uri`端点的路径。

> **Note**
>
> URL 应该以 `https://` 开头，并且不应以下斜杠 / 结尾。只有端口号 443 和 80 可用于此 URL。例如，Salesforce 使用以下 URL：`https://login.salesforce.com`
> 如果选择自动填充，则发现文档必须对以下值使用 HTTPS：`authorization_endpoint`、`token_endpoint`、`userinfo_endpoint` 和 `jwks_uri`。否则，登录将失败。

12. 在 OpenID Connect 提供商和用户池之间的 “映射属性” 下配置属性映射规则。用户池属性是 Amazon Cognito 用户个人资料中的目标属性，OpenID Conn ect 属性是你希望 Amazon Cognito 在 ID-Token 索赔或响应中找到的来源属性。`userInfo`Amazon Cognito 会自动将 OIDC 索赔订阅映射到目标用户个人`username`资料中。

    有关更多信息，请参阅 将 IdP 属性映射到配置文件和令牌。
13. 选择添加身份提供者。
14. 从 “应用程序客户端” 菜单中，从列表中选择一个应用程序客户端。导航到 “登录页面” 选项卡，在 “托管登录页面配置” 下，选择 “编辑”。找到身份提供者并添加您的新 OIDC IdP。
15. 选择保存更改。

API/CLI

请参阅示例二中的 OIDC 配置，网址为。CreateIdentityProvider您可以修改此语法并将其用作`CreateIdentityProvider``UpdateIdentityProvider`、的请求正文或的`--cli-input-json`输入文件create-identity-provider。

测试 OIDC IdP 配置

在您的应用程序中，您必须在用户的客户端中调用浏览器，这样他们才能使用其 OIDC 提供商登录。完成前几节中的设置过程后，请向您的提供商测试登录。以下示例网址会加载带有前缀域名的用户池的登录页面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://www.example.com
```

当您进入应用程序客户端菜单，选择应用程序客户端，导航到登录页面选项卡，然后选择查看登录页面时，Amazon Cognito 会引导您进入该页面。有关用户池域的更多信息，请参阅 配置用户池域。有关应用程序客户端（包括客户端 IDs 和回调）的更多信息 URLs，请参阅特定于应用程序的应用程序客户端设置。

以下示例链接对端点授权使用`identity_provider`查询参数设置了从的静默重定向到`MyOIDCIdP`提供商。此 URL 绕过使用托管登录的交互式用户池登录，直接转到 IdP 登录页面。

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
identity_provider=MyOIDCIdP&response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com
```

OIDC 用户池 IdP 身份验证流程

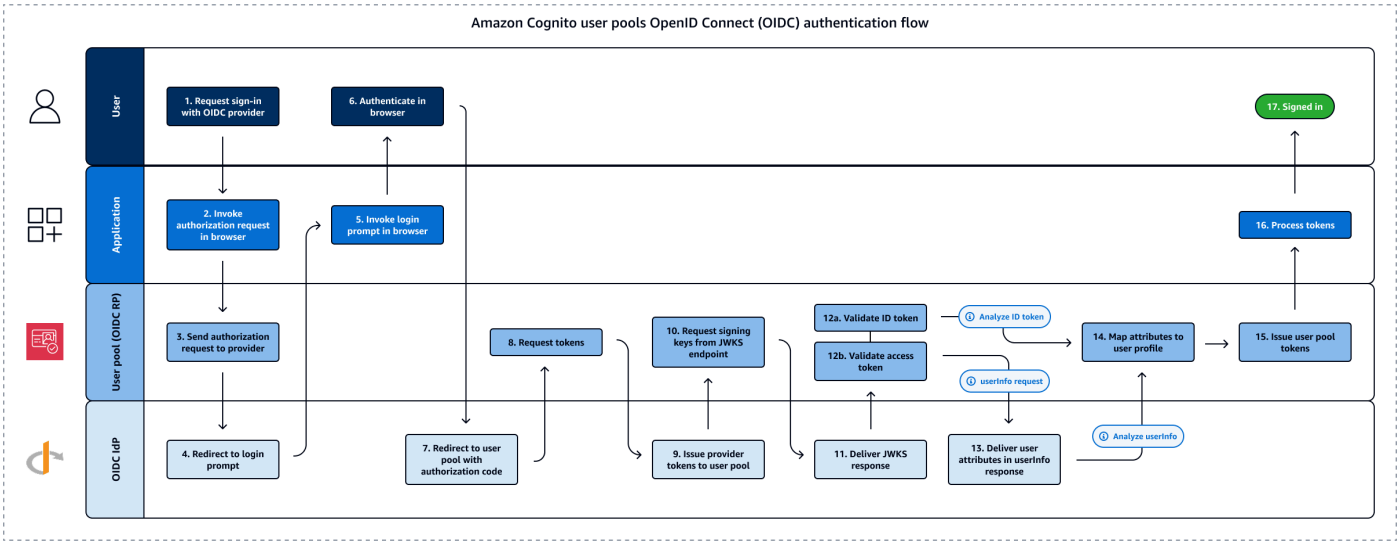
通过 OpenID Connect（OIDC）登录，您的用户池会自动执行身份提供者（IdP）的授权码登录流程。您的用户使用其 IdP 完成登录后，Amazon Cognito 会在外部提供商的 `oauth2/idpresponse` 端点收集他们的代码。借助生成的访问令牌，您的用户池查询 IdP `userInfo` 端点以检索用户属性。然后，您的用户池将收到的属性与您设置的属性映射规则进行比较，并相应地填入用户的配置文件和 ID 令牌。

您在 OIDC 提供商配置中请求的 OAuth 2.0 范围定义了 IdP 向 Amazon Cognito 提供的用户属性。作为一项最佳安全实践，请仅请求与要映射到用户池的属性相对应的范围。例如，如果您的用户池请求 `openid profile`，您将获得所有可能的属性，但是如果您请求 `openid email phone_number`，则只能获得用户的电子邮件地址和电话号码。您可以将向 OIDC 请求的范围配置 IdPs为与您在应用程序客户端和用户池身份验证请求中授权和请求的范围不同。

当您的用户使用 OIDC IdP 登录您的应用程序时，您的用户池执行以下身份验证流程。

1. 用户访问您的托管登录登录页面，并选择使用其 OIDC IdP 登录。
2. 您的应用程序将用户的浏览器定向到用户池的授权端点。
3. 您的用户池将请求重定向到 OIDC IdP 的授权端点。
4. 您的 IdP 会显示登录提示。
5. 在您的应用程序中，您的用户会话显示 OIDC IdP 的登录提示。
6. 用户输入他们的 IdP 凭证，或者为已通过身份验证的会话提供 cookie。
7. 在您的用户进行身份验证后，OIDC IdP 将使用授权代码重定向至 Amazon Cognito。

8. 您的用户池使用授权码兑换 ID 和访问令牌。当你为你的 IdP 配置作用域时，Amazon Cognito 会收到访问令牌。openidID 令牌中的声明和userInfo响应由您的 IdP 配置中的其他范围决定，例如profile和。email
9. 您的 IdP 会发放所请求的代币。
10. 您的用户池 URLs 在您的 IdP 配置中确定颁发者到 IdP jwks_uri 终端节点的路径，并从 JSON 网络密钥集 (JWKS) 端点请求令牌签名密钥。
11. IdP 从 JWKS 端点返回签名密钥。
12. 您的用户池根据令牌中的签名和到期数据验证 IdP 令牌。
13. 您的用户池使用访问令牌向 userInfo IdP 终端节点授权请求。IdP 根据访问令牌范围使用用户数据进行响应。
14. 您的用户池将 ID 令牌和 IdP 的userInfo响应与用户池中的属性映射规则进行比较。它将映射的 IdP 属性写入用户池配置文件属性。
15. Amazon Cognito 颁发应用程序持有者令牌，可能包括身份令牌、访问令牌和刷新令牌。
16. 您的应用程序处理用户池令牌并让用户登录。



> **Note**
>
> Amazon Cognito 会取消未在 5 分钟内完成的身份验证请求，并将用户重定向到托管登录。页面随即显示 Something went wrong 错误消息。

OIDC 是 OAuth 2.0 之上的身份层，它指定由 IdPs OIDC 客户端应用程序（依赖方）颁发的 JSON 格式 (JWT) 身份令牌。有关将 Amazon Cognito 添加为 OIDC 信赖方的信息，请参阅适用于您 OIDC IdP 的文档。

当用户使用授权码授予进行身份验证时，用户群体将返回 ID、访问权限和刷新令牌。ID 令牌是用于身份管理的标准 OIDC 令牌，访问令牌是标准 OAuth 2.0 令牌。有关您的用户群体应用程序客户端可以支持的授权类型的更多信息，请参阅对端点授权。

用户群体如何处理来自 OIDC 提供者的声明

当您的用户通过第三方 OIDC 提供商完成登录时，托管登录会从 IdP 检索授权码。用户群体会与 IdP 的 `token` 端点交换访问令牌和 ID 令牌的授权码。用户群体不会将这些令牌传递给用户或应用程序，而是使用它们来构建用户配置文件，其中包含用户群体在声明中以其自己的令牌表示的数据。

Amazon Cognito 不会独立验证访问令牌。相反，它会从提供者 `userInfo` 端点请求用户属性信息，如果令牌无效，则该请求会被拒绝。

Amazon Cognito 通过以下检查来验证提供者 ID 令牌：

1. 检查提供者是否使用以下集合中的算法对令牌进行了签名：RSA、HMAC、椭圆曲线。
2. 如果提供者使用非对称签名算法对令牌进行了签名，请检查令牌 `kid` 声明中的签名密钥 ID 是否在提供者 `jwks_uri` 端点上列出。Amazon Cognito 会为其处理的每个 IdP ID 令牌刷新来自你的 IdP 配置中的 JWKS 终端节点的签名密钥。
3. 根据提供者元数据，将 ID 令牌签名与预期的签名进行比较。
4. 将 `iss` 声明与为 IdP 配置的 OIDC 颁发者进行比较。
5. 比较 `aud` 声明是否与在 IdP 上配置的客户端 ID 相匹配，或者，如果 `aud` 声明中有多个值，则声明包含所配置的客户端 ID。
6. 检查 `exp` 声明中的时间戳不早于当前时间。

用户群体会验证 ID 令牌，然后尝试使用提供者访问令牌向提供者 `userInfo` 端点发出请求。此请求检索访问令牌中的范围授权它读取的任何用户配置文件信息。然后，用户群体将搜索您在用户群体中根据需要设置的用户属性。您必须在提供者配置中为必需的属性创建属性映射。用户群体会检查提供者 ID 令牌和 `userInfo` 响应。用户群体将所有与映射规则匹配的声明写入用户群体用户配置文件中的用户属性。用户群体会忽略与映射规则匹配、但不是必需且在提供者的声明中找不到的属性。

将 IdP 属性映射到配置文件和令牌

身份提供者（IdP）服务（包括 Amazon Cognito）通常可以记录有关用户的更多信息。您可能想知道他们在哪家公司工作、如何联系他们以及其他可供识别的信息。但是，这些属性的格式在不同的提供者之间存在差异。例如，使用您的用户池设置 IdPs 来自三个不同供应商的三个供应商，并检查每个供应商的 SAML 断言、ID 令牌或`userInfo`负载示例。其中一个 IdP 将用户的电子邮件地址表示为 `email`，另一个表示为 `emailaddress`，第三个表示为 `http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`。

IdPs 与用户池整合带来的一个主要好处是能够将各种属性名称映射到具有一致、可预测的共享属性名称的单个 OIDC 令牌架构中。这样，您的开发人员就无需维护处理各种复杂单点登录事件的逻辑。这种格式整合就是属性映射。用户池属性映射将 IdP 属性名称分配给相应的用户池属性名称。例如，您可以将用户池配置为将 `emailaddress` 声明的值写入标准用户池属性 `email`。

每个用户池 IdP 都有单独的属性映射架构。要为您的 IdP 指定属性映射，请在 Amazon Cognito 控制台、 Amazon SDK 或用户池 REST API 中配置用户池身份提供者。

有关映射的需知信息

在开始设置用户属性映射之前，请查看以下重要详细信息。

- 当联合用户登录到您的应用程序时，您的用户群体需要的每个用户群体属性都必须存在一个映射。例如，如果您的用户群体需要 `email` 属性来进行注册，则将此属性映射到 IdP 中的对等属性。
- 默认情况下，映射的电子邮件地址未经验证。您无法使用一次性代码验证映射的电子邮件地址。而是要映射 IdP 的属性来获取验证状态。例如，Google 和大多数 OIDC 提供商都包含 `email_verified` 属性。
- 您可以将身份提供者（IdP）令牌映射到用户群体中的自定义属性。社交提供者提供访问令牌，而 OIDC 提供者提供访问令牌和 ID 令牌。要映射令牌，请添加一个最大长度为 2048 个字符的自定义属性，向您的应用程序客户端授予对该属性的写入权限，然后将 `access_token` 或 `id_token` 从 IdP 映射到自定义属性。
- 对于每个映射的用户群体属性，最大值长度 2048 个字符必须足够大，才能容纳 Amazon Cognito 从 IdP 处获取的值。否则，当用户登录到您的应用程序时，Amazon Cognito 会报告错误。当令牌长度超过 2048 个字符时，Amazon Cognito 不支持将 IdP 令牌映射到自定义属性。
- Amazon Cognito 从联合身份验证 IdP 传递的特定声明中派生出联合用户配置文件中的 `username` 属性，如下表所示。Amazon Cognito 在此属性值前面加上 IdP 的名称，例如 `MyOIDCIdP_[sub]`。当您希望联合用户拥有与外部用户目录中的属性完全匹配的属性时，请将该属性映射到 Amazon Cognito 登录属性，如 `preferred_username`。

身份提供商	**username** 资源属性
Facebook	`id`
Google	`sub`
Login with Amazon	`user_id`
通过 Apple 登录	`sub`
SAML 提供商	`NameID`
OpenID Connect (OIDC) 提供者	`sub`

- 当用户池不区分大小写时，Amazon Cognito 会将联合用户自动生成的用户名中的用户名源属性转换为小写字母。以下是区分大小写的用户池的示例用户名：`MySAML_TestUser@example.com`。以下是不区分大小写的用户池的相同用户名：`MySAML_testuser@example.com`。

  在不区分大小写的用户池中，处理用户名的 Lambda 触发器必须考虑到对任何混合大小写的声明进行的这种修改，以适应用户名称源属性。要将您的 IdP 链接到区分大小写设置与当前用户池不同的用户池，请创建一个新的用户池。

- 当用户登录您的应用程序时，Amazon Cognito 必须能够更新映射的用户池属性。用户通过某个 IdP 登录时，Amazon Cognito 将使用来自该 IdP 的最新信息更新映射的属性。Amazon Cognito 将更新每个映射的属性，即使当前值已经与最新信息匹配也会更新。要确保 Amazon Cognito 可以更新属性，请检查以下要求：

  - 您从 IdP 映射的所有用户群体自定义属性都必须为可变的。您可以随时更新可变的自定义属性。相比之下，只有在首次创建用户配置文件时，才能为用户的不可变 自定义属性设置值。要在 Amazon Cognito 控制台中创建可变的自定义属性，请在注册菜单中选择添加自定义属性时为添加的属性激活可变复选框。或者，如果您使用 CreateUserPoolAPI 操作创建用户池，则可以将每个属性的`Mutable`参数设置为`true`。如果您的 IdP 为映射的不可变属性发送一个值，Amazon Cognito 会返回错误且登录失败。

  - 在应用程序的应用程序客户端设置中，映射的属性必须可写。您可以在 Amazon Cognito 控制台的 App clients (应用程序客户端) 页面中设置哪些属性为可写属性。或者，如果您使用 `CreateUserPoolClient` API 操作创建应用程序客户端，则可以将这些属性添加到 `WriteAttributes` 数组。如果您的 IdP 为映射的不可写属性发送一个值，Amazon Cognito 不会设置该属性值，而是继续进行身份验证。

- 当 IdP 属性包含多个值时，Amazon Cognito 会将所有值扁平化为一个以逗号分隔的字符串，并用方括号字符和。[ ]Amazon Cognito URL 对包含除了、、和之外的非字母数字字符的值进行表单编码。. - * _在您的应用程序中使用这些值之前，您必须解码并解析各个值。

指定适用于用户池的身份提供商属性映射（Amazon Web Services Management Console）

您可以使用为您的用户 Amazon Web Services Management Console 池的 IdP 指定属性映射。

> **Note**
>
> 只有当陈述存在于传入令牌中时，Amazon Cognito 才会将传入陈述映射到用户池属性。如果之前映射的声明不再存在于传入令牌中，则不会被删除或更改。如果您的应用程序需要映射已删除的声明，则可以使用预身份验证 Lambda 触发器在身份验证期间删除自定义属性，并允许从传入令牌重新填充这些属性。

指定社交 IdP 属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 选择 “社交和外部提供商” 菜单。
4. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的 Facebook、Google、Amazon 或 Apple IdP。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。

   有关添加社交 IdP 的更多信息，请参阅将社交身份提供者与用户池配合使用。
5. 对于需要映射的每个属性，请完成以下步骤：
   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。
   b. 从属性列中选择一个`<provider>`属性。这将是从提供商目录传递的属性。在下拉列表中提供来自社交服务提供商的已知属性。
   c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。
6. 选择 Save changes（保存更改）。

指定 SAML 提供商属性映射

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 选择 “社交和外部提供商” 菜单。
4. 选择 Add an identity provider（添加身份提供商），或者选择您已配置的 SAML IdP。找到 Attribute mapping（属性映射），然后选择 Edit（编辑）。有关添加 SAML IdP 的更多信息，请参阅将 SAML 身份提供者与用户池配合使用。
5. 对于需要映射的每个属性，请完成以下步骤：

   a. 从 User pool attribute（用户池属性）列中选择属性。这是分配给您的用户池中用户配置文件的属性。自定义属性在标准属性之后列出。

   b. 从 SAML attribute（SAML 属性）列中选择属性。这将是从提供商目录传递的属性。

      您的 IdP 可能会提供示例 SAML 断言以供参考。有些 IdPs 使用简单的名称，例如`email`，而另一些则使用类似于 URL 格式的属性名称：

      ```
      http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress
      ```

   c. 要在 IdP 和 Amazon Cognito 之间映射其它属性，请选择 Add another attribute（添加其它属性）。
6. 选择保存更改。

为您的用户池（Amazon CLI 和 Amazon API）指定身份提供商属性映射

以下请求正文按顺序将 SAML 提供商 “MyIdP” 属性`emailaddressbirthdate`、、和UpdateIdentityProvider映射`phone`到用户池属性`emailbirthdatephone_number`、和。CreateIdentityProvider这是 SAML 2.0 提供者的完整请求正文，您的请求正文将因 IdP 类型和具体细节而有所不同。属性映射位于 `AttributeMapping` 参数中。

```
{
   "AttributeMapping": {
      "email" : "emailaddress",
      "birthdate" : "birthdate",
      "phone_number" : "phone"
   },
   "IdpIdentifiers": [
      "IdP1",
```

```
        "pdxsaml"
    ],
    "ProviderDetails": {
        "IDPInit": "true",
        "IDPSignout": "true",
        "EncryptedResponses" : "true",
        "MetadataURL": "https://auth.example.com/sso/saml/metadata",
        "RequestSigningAlgorithm": "rsa-sha256"
    },
    "ProviderName": "MyIdP",
    "ProviderType": "SAML",
    "UserPoolId": "us-west-2_EXAMPLE"
}
```

使用以下命令为您的用户池指定 IdP 属性映射。

在提供商创建时指定属性映射

- Amazon CLI: `aws cognito-idp create-identity-provider`

  带元数据文件的示例：`aws cognito-idp create-identity-provider --user-pool-id <user_pool_id> --provider-name=SAML_provider_1 --provider-type SAML --provider-details file:///details.json --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`

  其中 `details.json` 包含：

  ```
  {
      "MetadataFile": "<SAML metadata XML>"
  }
  ```

  > **Note**
  >
  > 如果*<SAML metadata XML>*包含任何引号 (")，则必须对其进行转义 (\")。

  带元数据 URL 的示例：

  ```
  aws cognito-idp create-identity-provider \
  --user-pool-id us-east-1_EXAMPLE \
  --provider-name=SAML_provider_1 \
```

```
--provider-type SAML \
--provider-details MetadataURL=https://myidp.example.com/saml/metadata \
--attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/
emailaddress
```

- API/SDK：CreateIdentityProvider

指定现有 IdP 的属性映射

- Amazon CLI: `aws cognito-idp update-identity-provider`

  示例：`aws cognito-idp update-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name> --attribute-mapping email=http://schemas.xmlsoap.org/ws/2005/05/identity/claims/emailaddress`
- API/SDK：UpdateIdentityProvider

获取有关特定 IdP 的属性映射的信息

- Amazon CLI: `aws cognito-idp describe-identity-provider`

  示例：`aws cognito-idp describe-identity-provider --user-pool-id <user_pool_id> --provider-name <provider_name>`
- API/SDK：DescribeIdentityProvider

将联合用户与现有用户配置文件关联

通常，同一个用户的个人资料包含多个身份提供商 (IdPs)，您已将其连接到您的用户池。Amazon Cognito 可以将用户的每次出现与目录中的同一个用户配置文件关联。这样，一个拥有多个 IdP 用户的人就可以在您的应用中获得一致的体验。 AdminLinkProviderForUser让 Amazon Cognito 将您的联合目录中用户的唯一 ID 识别为用户池中的用户。当您具有零个或多个与用户配置文件相关联的联合身份时，用户群体中的一个用户将计为一个每月活跃用户 (MAU) 以进行计费。

当联合用户首次登录您的用户群体时，Amazon Cognito 会查找您已关联到其身份的本地个人资料。如果不存在关联的个人资料，则您的用户群体会创建一个新的个人资料。您可以在联合用户首次登录前的任何时候，通过计划中的预先准备任务或 注册前 Lambda 触发器 中的 `AdminLinkProviderForUser` API 请求，创建本地配置文件将其链接到联合用户。在您的用户登录且 Amazon Cognito 检测到关联的本地个人资料后，您的用户群体会读取用户的声明，并将其与 IdP 的映射规则进行比较。然后，您的用户群体会使用他们登录时映射的声明来更新

关联的本地个人资料。通过这种方式，您可以为本地配置文件配置访问权限声明，并将其身份声明保留给您的 up-to-date提供商。在 Amazon Cognito 将您的联合用户与关联的个人资料匹配后，他们将始终登录该个人资料。然后，您可以将用户的更多提供者身份关联到同一个人资料，从而为一位客户提供一致的应用体验。要关联之前登录过的联合用户，必须先删除其现有个人资料。您可以通过其格式识别现有个人资料：`[Provider name]_identifier`。例如 `LoginWithAmazon_amzn1.account.AFAEXAMPLE`。您创建并链接到第三方用户身份的用户具有创建时使用的用户名和包含其关联身份详细信息的 `identities` 属性。

> **Important**
>
> 由于`AdminLinkProviderForUser`允许具有外部联合身份的用户以用户池中的现有用户身份登录，因此必须仅将其与应用程序所有者信任的外部 IdPs 和提供者属性一起使用。

例如，如果您是托管服务提供商 (MSP)，具有一个与多个客户共享的应用程序。每个客户都通过 Active Directory Federation Services (ADFS) 登录您的应用程序。您的 IT 管理员 Carlos 在每个客户的域中都有一个账户。您希望 Carlos 在每次登录时都能被识别为应用程序管理员，而不考虑 IdP。

你的 ADFS 在 IdPs 向亚马逊 Cognito 提出的 S `msp_carlos@example.com` AML `email` 声明中出示了 Carlos 的电子邮件地址。您使用用户名 `Carlos` 在用户群体中创建一个用户。以下 Amazon Command Line Interface (Amazon CLI) 命令将来自 IdPs ADFS1、 ADFS2和 Carlos 的身份链接起来。 ADFS3

> **Note**
>
> 您可以根据特定属性声明关联用户。这种能力是 OIDC 和 SAML 所独有的。 IdPs 对于其他提供者类型，您必须基于固定来源属性来进行关联。有关更多信息，请参阅 AdminLinkProviderForUser。当您将社交 IdP 与用户配置文件关联时，必须将 `ProviderAttributeName` 设置为 `Cognito_Subject`。`ProviderAttributeValue` 必须是用户在 IdP 中的唯一标识符。

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS1,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

```
aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS2,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com

aws cognito-idp admin-link-provider-for-user \
--user-pool-id us-east-1_EXAMPLE \
--destination-user ProviderAttributeValue=Carlos,ProviderName=Cognito \
--source-user
 ProviderName=ADFS3,ProviderAttributeName=email,ProviderAttributeValue=msp_carlos@example.com
```

用户群体中的用户配置文件 `Carlos` 现在具有以下 `identities` 属性。

```
[{
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS1",
    "providerType": "SAML",
    "issuer": "http://auth.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS2",
    "providerType": "SAML",
    "issuer": "http://auth2.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}, {
    "userId": "msp_carlos@example.com",
    "providerName": "ADFS3",
    "providerType": "SAML",
    "issuer": "http://auth3.example.com",
    "primary": false,
    "dateCreated": 111111111111111
}]
```

有关关联联合用户需了解的事项

- 您最多可以将五个联合用户与每个用户配置文件关联。
- 根据 `AdminLinkProviderForUser` API 请求中的 `SourceUser` 的 `ProviderAttributeName` 参数的定义，您最多可以将用户链接到五个 IdP 属性声明中的每个 IdP。例如，如果您已将至少一个

用户链接到源属性 email、phone、department、given_name 和 location，则只能选择这五个属性中的一个来链接额外的用户。

- 您可以将联合用户与现有联合用户配置文件或本地用户关联。
- 您无法在中将提供商链接到用户个人资料 Amazon Web Services Management Console。
- 您的用户 ID 令牌在 identities 声明中含包所有关联的提供者。
- 您可以在 API 请求中为自动创建的联合用户配置文件设置密码。AdminSetUserPassword然后，该用户的状态从 EXTERNAL_PROVIDER 更改为 CONFIRMED。处于此状态的用户能够以联合用户身份登录，并像关联的本地用户一样在 API 中启动身份验证流程。他们还可以在经过令牌验证的 API 请求中修改自己的密码和属性，例如和。ChangePasswordUpdateUserAttributes作为最佳安全实践，为了让用户与您的外部 IdP 保持同步，请勿在联合用户配置文件上设置密码。相反，使用 AdminLinkProviderForUser 将用户与本地配置文件关联。
- 当用户通过 IdP 进行登录时，Amazon Cognito 会将用户属性填充到关联的本地用户配置文件中。Amazon Cognito 处理来自 OIDC IdP 的 ID 令牌中的身份声明，还会检查 2.0 和 OIDC 提供商userInfo的 OAuth终端节点。Amazon Cognito 将 ID 令牌中的信息优先级置于来自 userInfo 的信息之上。

当您得知用户不再使用您已关联到其个人资料的外部用户账户时，您可以取消该用户账户与您的用户群体用户的关联。当您关联用户时，您会在请求中提供用户的属性名称、属性值和提供者名称。要删除用户不再需要的配置文件，请使用等效参数发出 AdminDisableProviderForUserAPI 请求。

AdminLinkProviderForUser有关其他命令语法和示例，请参见 Amazon SDKs。

用户池托管登录

您可以选择一个网域来托管用户池的服务。当您添加域名时，Amazon Cognito 用户池将获得以下功能，这些功能统称为托管登录。

- 一种授权服务器，充当使用 OAuth 2.0 和 OpenID Connect (OIDC) 的应用程序的身份提供者 (IdP)。授权服务器路由请求、发出和管理 JSON Web 令牌 (JWTs)，并提供用户属性信息。
- 用于登录、注销和密码管理等身份验证操作的 ready-to-use用户界面 (UI)。托管登录页面充当身份验证服务的 Web 前端。
- SAML 2.0、OIDC、Facebook、Lo IdPs gin with Amazon IdPs、使用苹果登录和谷歌的服务提供商 (SP) 或依赖方 (RP)。

与托管登录共享某些功能的另一个选项是经典的托管用户界面。经典托管用户界面是托管登录服务的第一代版本。托管 UI IdP 和 RP 服务通常与托管登录具有相同的特征，但是登录页面的设计更简单，功能也更少。例如，密钥登录在经典托管用户界面中不可用。在精简版功能计划中，经典托管用户界面是用户池域服务的唯一选择。

托管登录页面是一系列 Web 界面，用于用户池中的基本注册、登录、多因素身份验证和密码重置活动。当你想让用户选择登录选项时，它们还会将用户连接到一个或多个第三方身份提供商 (IdPs)。当您想要对用户进行身份验证和授权时，您的应用程序可以在用户的浏览器中调用您的托管登录页面。

您可以通过自定义徽标、背景和样式使托管登录用户体验与您的品牌相匹配。您可以为托管登录用户界面设置两个品牌选项：用于托管登录的品牌编辑器和托管用户界面的托管用户界面（经典）品牌。

品牌编辑器

Amazon Cognito 控制台中提供了最多的 up-to-date身份验证选项和可视化编辑器，带来了更新的用户体验。

托管用户界面品牌

对于以前使用 Amazon Cognito 用户池的人来说，这是一种熟悉的用户体验。托管 UI 的品牌推广是基于文件的系统。要将品牌应用于托管的用户界面页面，您需要上传徽标图像文件和一个为多个预定的 CSS 样式选项设置值的文件。

品牌编辑器并非在用户池的所有功能计划中都可用。有关更多信息，请参阅 用户池功能计划。

有关构造对托管登录和托管用户界面服务的请求的更多信息，请参阅用户池端点和托管登录参考。

> **Note**
>
> Amazon Cognito 托管登录不支持使用自定义身份验证质询 Lambda 触发器的自定义身份验证。

主题

- 托管登录本地化
- 使用设置托管登录 Amazon Amplify
- 使用 Amazon Cognito 控制台设置托管登录
- 查看您的登录页面
- 自定义您的身份验证页面

- 关于托管登录和托管用户界面的注意事项
- 配置用户池域
- 将品牌应用于托管登录页面

托管登录本地化

在用户交互式页面中，托管登录默认为英语。您可以显示根据您选择的语言进行本地化的托管登录页面。可用的语言是中可用的语言 Amazon Web Services Management Console。在分发给用户的链接中，添加`lang`查询参数，如以下示例所示。

```
https://<your domain>/oauth2/authorize?lang=es&response_type=code&client_id=<your app
 client id>&redirect_uri=<your relying-party url>
```

Amazon Cognito 在用户的浏览器中设置一个 cookie，并在用户使用参数进行初始请求后，使用他们的语言偏好。`lang`设置 Cookie 后，语言选择将保持不变，无需在请求中显示或要求您在请求中包含该参数。例如，用户使用`lang=de`参数发出登录请求后，他们的托管登录页面将以德语呈现，直到他们清除 Cookie 或使用新的本地化参数（例如`lang=en`）提出新请求。

本地化仅适用于托管登录。您必须使用 Essentials 或 Plus 功能套餐并已分配域名才能使用托管登录品牌。

您的用户在托管登录中所做的选择不适用于自定义电子邮件或短信发送器触发器。实现这些触发器时，必须使用其他机制（例如`locale`属性）来确定用户的首选语言。

有以下语言可供选择。

托管登录语言

语言	代码
德语	de
English	en
西班牙语	es
法语	fr
印度尼西亚语	id

语言	代码
意大利语	它
日语	ja
韩语	ko
葡萄牙语（巴西）	pt-BR
中文（简体）	zh-CN
中文（繁体）	zh-TW

使用设置托管登录 Amazon Amplify

如果您使用 Amazon Amplify 向网络或移动应用程序添加身份验证，则可以在 Amplify 命令行界面 (CLI) 中设置托管登录页面，在 Amplify 框架中设置库。要向您的应用程序添加身份验证，请将`Auth`类别添加到您的项目中。然后，在您的应用程序中，使用 Amplify 客户端库对用户池用户进行身份验证。

您可以调用托管登录页面进行身份验证，也可以通过重定向到 IdP 的授权端点联合用户。用户成功通过提供商进行身份验证后，Amplify 会在您的用户池中创建一个新用户，并将该用户的令牌传递给您的应用程序。

以下示例说明 Amazon Amplify 如何使用在您的应用程序中设置社交服务提供商的托管登录。

- Amazon Amplify 的身份验证 JavaScript。
- Amazon Amplify Swift 的身份验证。
- Amazon Amplify Flutter 的身份验证。
- Amazon Amplify 安卓版身份验证。

使用 Amazon Cognito 控制台设置托管登录

托管登录和托管 UI 的首要要求是用户池域。在用户池控制台中，导航到用户池的域选项卡，然后添加 Cognito 域或自定义域。您也可以在创建新用户池的过程中选择域。有关更多信息，请参阅 配置用户池域。当您的用户池中的某个域处于活动状态时，所有应用程序客户端都会在该域上提供公共身份验证页面。

创建或修改用户池域时，需要为域名设置品牌版本。此品牌版本可以选择托管登录或托管用户界面（经典）。您选择的品牌版本适用于在您的网域中使用登录服务的所有应用程序客户端。

下一步是从用户池的 “应用程序客户端” 选项卡中创建应用程序客户端。在创建应用程序客户端的过程中，Amazon Cognito 会要求您提供有关您的应用程序的信息，然后提示您选择返回网址。返回网址也称为信赖方 (RP) 网址、重定向 URI 和回调网址。这是您的应用程序运行时所使用的 URL，例如`https://www.example.com`或`myapp://example`。

在用户池中配置具有品牌风格的域和应用程序客户端后，您的托管登录页面就可以在互联网上使用。

查看您的登录页面

在 Amazon Cognito 控制台中，在应用程序客户端菜单下选择应用程序客户端登录页面选项卡中的查看登录页面按钮。此按钮将带您进入用户池域中的登录页面，其中包含以下基本参数。

- 应用程序客户端 id
- 授权代码授予请求
- 对您为当前应用程序客户端激活的所有范围的请求
- 当前应用程序客户端列表中的第一个回调 URL

当您想要测试托管登录页面的基本功能时，“查看登录页面” 按钮非常有用。您的登录页面将与您分配给用户池域名的品牌版本相匹配。您可以使用其他和修改后的参数自定义登录 URL。在大多数情况下，“查看登录页面” 链接中自动生成的参数并不完全符合您的应用程序的需求。在这些情况下，您必须自定义应用程序在用户登录时调用的 URL。有关登录参数键和值的更多信息，请参阅 用户池端点和托管登录参考。

登录网页使用以下 URL 格式。此示例使用 `response_type=code` 参数请求授权码授予。

```
https://<your domain>/oauth2/authorize?response_type=code&client_id=<your app client
 id>&redirect_uri=<your relying-party url>
```

您可以从用户池的 “域” 菜单中查找您的用户池域字符串。在应用程序客户端菜单中，您可以识别应用程序客户端 IDs、其回调 URLs、允许的范围和其他配置。

当您使用自定义参数导航到 `/oauth2/authorize` 端点时，Amazon Cognito 会将您重定向到 `/oauth2/login` 端点，或者在您具有 `identity_provider` 或 `idp_identifier` 参数时，静默地将您重定向到 IdP 登录页面。

隐式拨款请求示例

您可以使用以下 URL 查看您的登录网页，以获取隐式代码授权 where `response_type=token`。成功登录后，Amazon Cognito 会将用户池令牌返回到您的 Web 浏览器的地址栏。

```
https://mydomain.auth.us-east-1.amazoncognito.com/authorize?
response_type=token&client_id=1example23456789&redirect_uri=https://
mydomain.example.com
```

身份令牌和访问令牌显示为附加到您的重定向 URL 的参数。

以下内容是来自隐式授予请求的示例响应。

```
https://auth.example.com/
#id_token=eyJraaBcDeF1234567890&access_token=eyJraGhIjKlM1112131415&expires_in=3600&token_type=
```

自定义您的身份验证页面

过去，Amazon Cognito 仅使用经典托管用户界面托管登录页面，这是一种简单的设计，可为身份验证网页提供通用外观。您可以使用徽标图像自定义 Amazon Cognito 用户池，也可以使用指定某些 CSS 样式值的文件调整某些样式。后来，Amazon Cognito 推出了托管登录，这是一项更新的托管身份验证服务。品牌编辑器更新 look-and-feel了托管登录。品牌编辑器是一个无代码的可视化编辑器，其选项套件比托管用户界面自定义体验还要多。托管登录还引入了自定义背景图像和深色模式主题。

你可以在用户池中的托管登录和托管用户界面品牌体验之间切换。要了解有关自定义托管登录页面的更多信息，请参阅将品牌应用于托管登录页面。

关于托管登录和托管用户界面的注意事项

一小时的托管登录和托管的用户界面会话 cookie

当用户使用您的登录页面或第三方提供商登录时，Amazon Cognito 会在他们的浏览器中设置一个 Cookie。使用此 Cookie，用户可以在一小时内使用相同的身份验证方法再次登录。当他们使用浏览器 Cookie 登录时，他们会获得新的令牌，这些令牌持续时间在您的应用程序客户端配置中指定的持续时间内。用户属性或身份验证因素的更改不会影响他们使用浏览器 Cookie 再次登录的能力。

使用会话 Cookie 进行身份验证不会将 Cookie 持续时间重置为额外一小时。如果用户在最近一次成功进行交互式身份验证超过一个小时后尝试访问您的登录页面，则必须重新登录。

确认用户帐户并验证用户属性

对于用户池本地用户，当您将用户池配置为允许 Cognito 自动发送消息进行验证和确认时，托管登录和托管用户界面效果最佳。当您启用此设置时，Amazon Cognito 会向注册的用户发送一条包含确认码的消息。相反，当您确认用户为用户池管理员时，您的登录页面会在注册后显示一条错误消息。在这种状态下，Amazon Cognito 已创建新用户，但无法发送验证消息。您仍然可以确认用户为管理员，但他们可能会在遇到错误后联系您的支持部门。有关管理确认的更多信息，请参阅允许用户在您的应用程序中注册但以用户池管理员身份进行确认。

查看您对配置的更改

如果您对页面进行了样式更改，但页面没有立即显示，请等待几分钟，然后刷新页面。

解码用户池令牌

Amazon Cognito 用户池令牌使用算法进行 RS256 签名。您可以使用解码和验证用户池令牌。Amazon Lambda请参阅解码和验证 Amazon Cognito J WT 令牌。 GitHub

Amazon WAF 网页 ACLs

您可以将用户池配置为使用 Amazon WAF Web 规则保护为您的登录页面提供服务的域名和授权服务器 ACLs。当前，只有当您的托管登录品牌版本为托管用户界面（经典）时，您配置的规则才适用于这些页面。有关更多信息，请参阅 关于 Amazon WAF 网络 ACLs 和亚马逊 Cognito 的注意事项。

TLS 版本

托管登录和托管用户界面页面在传输过程中需要加密。Amazon Cognito 提供的用户池域要求用户的浏览器协商的最低 TLS 版本为 1.2。自定义域支持 TLS 版本 1.2 的浏览器连接。托管用户界面（经典）不要求自定义域名使用 TLS 1.2，但较新的托管登录要求自定义域和前缀域均使用 TLS 版本 1.2。由于 Amazon Cognito 管理您的域服务的配置，因此您无法修改用户池域的 TLS 要求。

CORS 策略

托管登录和托管用户界面均不支持自定义跨域资源共享 (CORS) 来源策略。CORS 策略将阻止用户在其请求中传递身份验证参数。取而代之的是，在应用程序前端实施 CORS 策略。Amazon Cognito 会向以下终端节点发送请求返回`Access-Control-Allow-Origin: *`响应标头。

1. 令牌端点
2. 撤消端点
3. userInfo 端点

Cookie

托管登录和托管用户界面会在用户的浏览器中设置 Cookie。这些 Cookie 符合某些浏览器的要求，即要求网站不能设置第三方 Cookie。它们的范围仅限于您的用户池端点，包括以下内容：

- 每个请求有一个 `XSRF-TOKEN` Cookie。
- 一个 `csrf-state` Cookie，用于在重定向用户时保持会话一致性。
- 用于会话一致性的 `csrf-state-legacy` Cookie，当您的浏览器不支持该属性时，Amazon Cognito 会将其读取为后备选项。`SameSite`
- 一个 `cognito` 会话 Cookie，用于将成功的登录尝试保留一小时。
- 一种保留用户在托管登录中对语言本地化的选择的 `lang` Cookie。
- 一种在用户在托管登录页面之间导航时保留所需数据的 `page-data` Cookie。

在 iOS 中，您可以阻止所有 Cookie。此设置与托管登录或托管用户界面不兼容。要与可能启用此设置的用户合作，请使用 Amazon SDK 将用户池身份验证构建到原生 iOS 应用程序中。在这种情况下，您可以构建自己的会话存储，该存储不依赖 Cookie。

托管登录版本变更的影响

请考虑添加域名和设置托管登录版本的以下影响。

- 当您添加带有托管登录或托管用户界面（经典）品牌的前缀域时，登录页面最多可能需要 60 秒钟才可用。
- 当您添加带有托管登录或托管用户界面（经典）品牌的自定义域名时，最多可能需要五分钟才能显示登录页面。
- 当您更改域名的品牌版本时，最多可能需要四分钟，您的登录页面才会在新的品牌版本中可用。
- 当您在托管登录和托管用户界面（经典）品牌之间切换时，Amazon Cognito 不会维护用户会话。他们必须使用新界面重新登录。

默认样式

当您在中创建应用程序客户端时 Amazon Web Services Management Console，Amazon Cognito 会自动为您的应用程序客户端分配品牌风格。当您通过该CreateUserPoolClient操作以编程方式创建应用程序客户端时，不会创建任何品牌风格。在您使用CreateManagedLoginBranding请求创建应用程序客户端之前，托管登录不可用于使用 Amazon SDK 创建的应用程序客户端。

托管登录身份验证提示超时

Amazon Cognito 会取消未在五分钟内完成的身份验证请求，并将用户重定向到托管登录。页面随即显示 `Something went wrong` 错误消息。

配置用户池域

配置域是设置用户池的可选部分。用户池域托管用于用户身份验证、与第三方提供者的联合身份验证以及 OpenID Connect（OIDC）流的特征。它具有托管登录，这是用于注册、登录和密码恢复等密钥操作的预建界面。它还托管标准 OpenID Connect (OIDC) 端点，例如授权、用户信息和令牌，用于 machine-to-machine (M2M) 授权以及其他 OIDC 和 2.0 身份验证和授权流程。 OAuth

用户使用与您的用户池关联的域中的托管登录页面进行身份验证。您可以通过两种方式配置此域：您可以使用默认 Amazon Cognito 托管域，也可以配置自己拥有的自定义域。

自定义域选项在灵活性、安全性和控制性方面有更多选择。例如，一个熟悉的、由组织拥有的域可以增强用户信任并使登录过程更加直观。但是，自定义域名方法需要一些额外的开销，例如管理 SSL 证书和 DNS 配置的开销。

终端节点 URLs 和`/.well-known/jwks.json`令牌签名密钥`/.well-known/openid-configuration`的 OIDC 发现端点未托管在您的域中。有关更多信息，请参阅 身份提供者和依赖方端点。

了解如何为用户池配置和管理域是将身份验证集成到应用程序中的重要步骤。使用用户池 API 和 S Amazon DK 登录可以替代配置网域。基于 API 的模型直接在 API 响应中提供令牌，但是对于使用用户池扩展功能作为 OIDC IdP 的实现，必须配置域。有关用户池中可用的身份验证模型的更多信息，请参阅了解 API、OIDC 和托管登录页面身份验证。

主题

- 用户池域需知信息
- 使用 Amazon Cognito 前缀域进行托管登录
- 使用自己的域名进行托管登录

用户池域需知信息

用户池域是应用程序中的 OIDC 依赖方和 UI 元素的服务点。当您计划为用户池实施域时，请考虑以下详细信息。

保留术语

您不能在 Amazon Cognito 前缀域的名称中使用文本 `aws`、`amazon` 或 `cognito`。

发现端点位于另一个域中

用户池发现端点 `.well-known/openid-configuration` 和 `.well-known/jwks.json` 不在您的用户池自定义域或前缀域中。这些端点的路径如下所示。

- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/openid-configuration`
- `https://cognito-idp.Region.amazonaws.com/your user pool ID/.well-known/jwks.json`

域名变更的有效时间

Amazon Cognito 可能需要长达一分钟的时间才能启动或更新前缀域名的品牌版本。对自定义域的更改最多可能需要五分钟才能传播。新的自定义域名可能需要长达一小时才能传播。

同时使用自定义域和前缀域

您可以使用自定义域名和拥有的前缀域名来设置用户池 Amazon。由于用户池发现端点托管在另一个域中，因此它们仅为自定义域提供服务。例如，您的 `openid-configuration` 将为 `"authorization_endpoint"` 提供单个值：`"https://auth.example.com/oauth2/authorize"`。

当用户池中同时包含自定义域和前缀域时，您可以使用自定义域来启用 OIDC 提供程序的完整特征。具有此配置的用户池中的前缀域没有发现或 token-signing-key端点，因此应相应地使用。

首选自定义域名作为密钥的信赖方 ID

使用密钥设置用户池身份验证时，必须设置信赖方 (RP) ID。如果您有自定义域名和前缀域名，则只能将 RP ID 设置为自定义域名。要在 Amazon Cognito 控制台中将前缀域设置为 RP ID，请删除您的自定义域名或将前缀域的完全限定域名 (FQDN) 输入为第三方域。

不要在域层次结构的不同级别使用自定义域

您可以为不同的用户池配置独立的自定义域，并且这些域可以共享同一个顶级域（TLD），例如 auth.example.com 和 auth2.example.com。托管登录会话 Cookie 对自定义域名和所有子域名有效，例如 * .auth.example.com。因此，您的应用程序的任何用户都不应访问任何父域和子域的托管登录。如果自定义域使用相同的 TLD，则应将其保持在相同的子域级别。

假设您有一个使用自定义域 auth.example.com 的用户池。然后创建另一个用户池并分配自定义域 uk.auth.example.com。用户使用 a uth.ex ample.com 登录。 并获得一个 Cookie，他们的浏览器会将其显示给通配符路径 * .auth.example.com 中的任何网站。然后他们尝试登录 uk.auth.example.com。

他们向您的用户池域传递了一个无效的 cookie，并收到错误消息，而不是登录提示。相比之下，拥有 *.auth.example.com 的 Cookie 的用户可以在 auth2.example.com 启动登录会话，而不会遇到任何问题。

品牌版本

创建域名时，您可以设置品牌版本。您可以选择更新的托管登录体验和经典的托管用户界面体验。此选项适用于在您的域中托管服务的所有应用程序客户端。

使用 Amazon Cognito 前缀域进行托管登录

托管登录的默认体验托管在 Amazon 拥有该域名的网域上。这种方法的进入门槛较低（选择一个前缀名称即可生效），但该方法没有自定义域所具有的值得信任的特征。Amazon Cognito 域选项和自定义域选项在费用上没有差异。唯一的差异是您引导用户访问的 Web 地址中的域。对于第三方 IdP 重定向和客户端凭证流，托管域几乎没有明显的影响。如果您的用户使用托管登录方式登录，并且会与与应用程序域不匹配的身份验证域进行交互，则自定义域更适合这种情况。

托管的 Amazon Cognito 域有您选择的前缀，但托管在根域 `amazoncognito.com`。以下是示例：

```
https://cognitoexample.auth.ap-south-1.amazoncognito.com
```

所有前缀域都遵循此格式:*prefix*. `auth`。 *Amazon Web Services ## code*。 `amazoncognito`。 `com`。自定义域用户池可以在您拥有的任何域上托管托管登录或托管用户界面页面。

> **Note**
>
> 为了增强您的 Amazon Cognito 应用程序的安全性，用户群体端点的父域将在公共后缀列表 (PSL) 中注册。PSL 可帮助用户的网络浏览器对您的用户群体端点及其设置的 Cookie 建立一致的理解。
> 用户池父域采用以下格式。
>
> ```
> auth.Region.amazoncognito.com
> auth-fips.Region.amazoncognito.com
> ```

要使用添加应用程序客户端和用户池域 Amazon Web Services Management Console，请参阅创建应用程序客户端。

主题

- 先决条件
- 配置 Amazon Cognito 域前缀
- 验证登录页面

先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅 用户池入门。

配置 Amazon Cognito 域前缀

您可以使用 Amazon Web Services Management Console 或 Amazon CLI 或 API 来配置用户池域。

Amazon Cognito console

配置域

1. 导航至 “品牌” 下的 “域名” 菜单。
2. 在 “域” 旁边，选择 “操作”，然后选择 “创建 Cognito 域”。如果您已经配置了用户池前缀域，请在创建新的自定义域之前选择 “删除 Cognito 域”。
3. 输入可用的域前缀，以与 Amazon Cognito 域结合使用。有关设置自定义域的信息，请参阅使用自己的域名进行托管登录。
4. 选择品牌推广版本。您的品牌版本适用于该域名的所有用户互动页面。您的用户池可以为所有应用程序客户端托管托管登录或托管用户界面品牌。

   > **Note**
   >
   > 您可以拥有自定义域和前缀域，但是 Amazon Cognito 仅为自定义域提供`/.well-known/openid-configuration`终端节点。

5. 选择创建。

CLI/API

使用以下命令可以创建域前缀并将其分配到您的用户池。

配置用户池域

- Amazon CLI: `aws cognito-idp create-user-pool-domain`

  示例：`aws cognito-idp create-user-pool-domain --user-pool-id <user_pool_id> --domain <domain_name> --managed-login-version 2`
- 用户池 API 操作：CreateUserPoolDomain

获取有关域的信息

- Amazon CLI: `aws cognito-idp describe-user-pool-domain`

  示例：`aws cognito-idp describe-user-pool-domain --domain <domain_name>`
- 用户池 API 操作：DescribeUserPoolDomain

删除域

- Amazon CLI: `aws cognito-idp delete-user-pool-domain`

  示例：`aws cognito-idp delete-user-pool-domain --domain <domain_name>`
- 用户池 API 操作：DeleteUserPoolDomain

验证登录页面

- 验证登录页面是否可从您的 Amazon Cognito 托管域访问。

```
https://<your_domain>/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

您的域显示在 Amazon Cognito 控制台的 Domain name (域名) 页面上。您的应用程序客户端 ID 和回调 URL 将显示在 App client settings（应用程序客户端设置）页面上。

使用自己的域名进行托管登录

设置应用程序客户端后，您可以使用托管登录的域服务的自定义域来配置用户池。使用自定义域名，用户可以使用您自己的网址而不是默认`amazoncognito.com`的前缀域名登录您的应用程序。自定义域

让您可以使用熟悉的域名，从而提高用户对您的应用程序的信任度，尤其是当根域与托管应用程序的域匹配时。自定义域名可以提高对组织安全要求的合规性。

自定义域有一些先决条件，包括用户池、应用程序客户端和您拥有的 Web 域。自定义域名还需要使用位于美国东部（弗吉尼亚北部）的 Amazon Certificate Manager (ACM) 管理的自定义域名的 SSL 证书。Amazon Cognito 会创建亚马逊 CloudFront 配送，在运输途中使用您的 ACM 证书进行保护。由于您拥有该域名，因此必须创建 DNS 记录，将流量引导至自定义域名的 CloudFront 分配。

在这些要素准备就绪后，您可以通过 Amazon Cognito 控制台或 API 将自定义域添加到用户池。这包括指定域名和 SSL 证书，然后使用提供的别名目标更新 DNS 配置。在进行这些更改后，您可以验证是否可从您的自定义域访问登录页面。

创建自定义域名最省力的方法是在 Amazon Route 53 中使用公共托管区域。Amazon Cognito 控制台只需几个步骤即可创建正确的 DNS 记录。在开始之前，请考虑为您拥有的域或子域创建 Route 53 托管区域。

主题

- 将自定义域添加到用户池
- 先决条件
- 步骤 1：输入自定义域名
- 步骤 2：添加别名目标和子域
- 步骤 3：验证登录页面
- 更改自定义域的 SSL 证书

将自定义域添加到用户池

要将自定义域添加到用户池，请在 Amazon Cognito 控制台中指定域名，并提供您使用 Amazon Certificate Manager（ACM）管理的证书。在添加域后，Amazon Cognito 提供了一个要添加到 DNS 配置的别名目标。

先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅 用户池入门。
- 您拥有的 Web 域。其父域必须具有有效的 DNS A 记录。您可以为该记录分配任何值。父域可以是域的根，也可以是域层次结构中上一级的子域。例如，如果您的自定义域是

auth.xyz.example.com，Amazon Cognito 必须能够将 xyz.example.com 解析为 IP 地址。为了防止对客户基础设施造成意外影响，Amazon Cognito 不支持使用顶级域名 (TLDs) 作为自定义域名。有关更多信息，请参阅域名。

- 能够为自定义域创建子域。我们建议对您的子域名进行身份验证。例如：*auth.example.com*。

> **Note**
>
> 如果您没有通配符证书，则可能需要为自定义域的子域获取新证书。

- 由美国东部（弗吉尼亚北部）的 ACM 管理的公共 SSL/TLS 证书。证书必须在 us-east-1 中，因为该证书将与全球服务中的发行版 CloudFront相关联。
- 支持服务器名称指示 (SNI) 的浏览器客户端。Amazon Cognito CloudFront 分配给自定义域名的分配需要 SNI。您无法更改此设置。有关 CloudFront 分发中的 SNI 的更多信息，请参阅《亚马逊 CloudFront 开发者指南》中的 “使用 SNI 处理 HTTPS 请求”。
- 一个应用程序，允许您的用户池授权服务器向用户会话中添加 Cookie。亚马逊 Cognito 为托管登录页面设置了几个必需的 Cookie。这包括 `cognito`、`cognito-fl` 和 `XSRF-TOKEN`。尽管每个 Cookie 都符合浏览器大小限制，但更改用户池配置可能会导致托管登录 Cookie 的大小增加。像自定义域前面的应用程序负载均衡器（ALB）这样的中间服务可能会强制设定最大请求头大小或总 Cookie 大小。如果您的应用程序还设置了自己的 Cookie，则用户的会话可能会超过这些限制。为避免大小限制冲突，我们建议您的应用程序不要在托管用户池域服务的子域上设置 Cookie。
- 允许更新 Amazon CloudFront 分配。为此，您可以为 Amazon Web Services 账户中的用户附加以下 IAM policy 声明：

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
            "Sid": "AllowCloudFrontUpdateDistribution",
            "Effect": "Allow",
            "Action": [
                "cloudfront:updateDistribution"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

有关授权操作的更多信息 CloudFront，请参阅使用基于身份的策略（IAM 策略）。 CloudFront

Amazon Cognito 最初使用您的 IAM 权限来配置 CloudFront分配，但分配由管理。 Amazon您无法更改 Amazon Cognito 与您的用户池关联的 CloudFront 分配的配置。例如，您无法更新安全策略中支持的 TLS 版本。

步骤 1：输入自定义域名

您可以使用 Amazon Cognito 控制台或 API 将域添加到用户池。

Amazon Cognito console

将域从 Amazon Cognito 控制台添加到用户池：

1. 导航至 “品牌” 下的 “域名” 菜单。
2. 选择域旁边的操作，然后选择创建自定义域或创建 Amazon Cognito 域。如果您已经配置了用户池自定义域，请在创建新的自定义域之前选择删除自定义域。
3. 在 “域” 旁边，选择 “操作”，然后选择 “创建自定义域”。如果您已经配置了自定义域名，请在创建新的自定义域名之前选择删除自定义域名以删除现有域名。
4. 对于自定义域，请输入您希望与 Amazon Cognito 一起使用的域的 URL。您的域名只能包含小写字母、数字和连字符。请勿对第一个或最后一个字符使用连字符。使用句点来分隔子域名。
5. 对于 ACM 证书，请选择要用于您的域的 SSL 证书。无论您的用户池如何，只有美国东部（弗吉尼亚北部）的 ACM 证书才有资格用于 Amazon Cognito 自定义域。 Amazon Web Services 区域

   如果您没有可用证书，则可以使用 ACM 在美国东部（弗吉尼亚北部）预置一个证书。有关更多信息，请参阅《Amazon Certificate Manager 用户指南》中的入门。
6. 选择品牌推广版本。您的品牌版本适用于该域名的所有用户互动页面。您的用户池可以为所有应用程序客户端托管托管登录或托管用户界面品牌。

   > **Note**
   > 您可以拥有自定义域和前缀域，但是 Amazon Cognito 仅为自定义域提供`/.well-known/openid-configuration`终端节点。

7. 选择创建。

8. 亚马逊 Cognito 会将您返回到域名菜单。此时将显示标题为在域名的 DNS 中创建别名记录的消息。记下控制台中显示的域和别名目标。在下一步骤中，将使用它们将流量指向您的自定义域。

API

以下CreateUserPoolDomain请求正文创建了一个自定义域。

```
{
   "Domain": "auth.example.com",
   "UserPoolId": "us-east-1_EXAMPLE",
   "ManagedLoginVersion": 2,
   "CustomDomainConfig": {
    "CertificateArn": "arn:aws:acm:us-east-1:111122223333:certificate/
a1b2c3d4-5678-90ab-cdef-EXAMPLE11111"
   }
}
```

步骤 2：添加别名目标和子域

在本步骤中，您将通过域名服务器（DNS）服务提供商设置一个别名，该别名指回到上一个步骤中的别名目标。如果您将 Amazon Route 53 用于 DNS 地址解析，请选择使用 Route 53 添加别名目标和子域部分。

将别名目标和子域添加到当前 DNS 配置

- 如果您没有将 Route 53 用于 DNS 地址解析，则必须使用您的 DNS 服务提供商配置工具将上一个步骤中的别名目标添加到域 DNS 记录中。您的 DNS 提供商还需要为您的自定义域设置子域。

使用 Route 53 添加别名目标和子域

1. 登录 Route 53 控制台。如果出现提示，请输入 Amazon 凭证。
2. 如果您在 Route 53 中没有公共托管区，请创建一个托管区，将根域作为您的自定义域的父域。有关更多信息，请参阅《Amazon Route 53 开发人员指南》中的创建公共托管区。
   a. 选择创建托管区域。
   b. 例如*auth.example.com*，从域名列表中输入自定义域名的父域名。*myapp.auth.example.com*

c. 输入托管区域的描述。

d. 选择公有托管区域的托管区域类型以允许公共客户端解析您的自定义域。不支持选择私有托管区域。

e. 根据需要应用标签。

f. 选择创建托管区域。

> **Note**
>
> 您还可以为自定义域创建新的托管区，并且使用父托管区中的委托集将查询指向子域托管区。否则，请创建 A 记录。此方法为您的托管区域提供了更大的灵活度和安全性。有关更多信息，请参阅通过 Amazon Route 53 为托管域创建子域。

3. 在托管区域页面上，选择您的托管区域的名称。

4. 为自定义域的父域添加 DNS 记录（如果您还没有 DNS 记录）。为父域创建 DNS 记录，其属性如下：

   - 记录名称：留空。
   - 记录类型：A。
   - 别名：不启用。
   - 值：输入您选择的目标。此记录必须解析到某个地址，但记录的值对于 Amazon Cognito 来说并不重要。
   - TTL：设置为您的首选 TTL 或保留为默认值。
   - 路由策略：选择简单路由。

5. 选择创建记录。以下是该域的示例记录*example.com*：

```
example.com. 60 IN A 198.51.100.1
```

> **Note**
>
> Amazon Cognito 验证您的自定义域的父域是否有 DNS 记录，以防止意外劫持生产域。如果您没有父域的 DNS 记录，则当您尝试设置自定义域时，Amazon Cognito 将返回错误消息。起始授权（SOA）记录是一条 DNS 记录，但不足以用于父域验证。

6. 使用以下属性为您的自定义域添加另一条 DNS 记录：

   - 记录名称：用于为 `auth.example.com` 创建记录的自定义域前缀（例如 `auth`）。

- 记录类型：A。
- 别名：启用。
- 流量路由至：选择 CloudFront 分配的别名。输入您之前记录的别名目标，例如 `123example.cloudfront.net`。
- 路由策略：选择简单路由。

7. 选择创建记录。

> **Note**
> 新记录可能需要大约 60 秒钟才能传播到所有 Route 53 DNS 服务器。您可以使用 Route 53 GetChangeAPI 方法来验证您的更改是否已传播。

步骤 3：验证登录页面

- 验证登录页面是否可从您的自定义域访问。

  通过在浏览器中输入此地址，使用您的自定义域和子域进行登录。以下是*example.com*带有子域名的自定义网域的网址示例：*auth*

```
https://myapp.auth.example.com/login?
response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callback_url>
```

更改自定义域的 SSL 证书

如果需要，您可以使用 Amazon Cognito 更改应用于自定义域的证书。

通常，在使用 ACM 进行常规证书续订后，此操作是不必要的。当您续订 ACM 中的现有证书时，证书的 ARN 保持不变，并且您的自定义域将自动使用新证书。

但是，如果您将现有证书替换为新证书，ACM 将为新证书提供一个新 ARN。要将新证书应用于自定义域，您必须将此 ARN 提供给 Amazon Cognito。

在提供新证书后，Amazon Cognito 需要长达 1 小时才能将它分配给自定义域。

> **开始前的准备工作**
>
> 您必须先将证书添加到 Amazon Cognito，然后才能更改 ACM 中的证书。有关更多信息，请参阅《Amazon Certificate Manager 用户指南》中的入门。
> 将您的证书添加到 ACM 时，您必须选择美国东部（弗吉尼亚北部）作为 Amazon 区域。

您可以使用 Amazon Cognito 控制台或 API 更改证书。

Amazon Web Services Management Console

更新 Amazon Cognito 控制台的证书：

1. 登录 Amazon Web Services Management Console 并打开 Amazon Cognito 控制台，网址为。https://console.amazonaws.cn/cognito/home
2. 选择用户池。
3. 选择要更新其证书的用户池。
4. 选择域名菜单。
5. 选择 操作、编辑 ACM 证书。
6. 选择您希望与自定义域关联的新证书。
7. 选择保存更改。

API

如要更新证书（Amazon Cognito API）

- 使用 UpdateUserPoolDomain 操作。

将品牌应用于托管登录页面

您可能需要在身份验证服务和应用程序之间提供一致的用户体验。您可以通过 Amazon SDK 中的自定义表单和后端 API 操作或托管登录来实现此目标。托管登录和经典托管用户界面是应用程序中提供用户池身份验证的组件的 Web 前端。要将托管身份验证服务与应用程序 UX 同步，您有两个自定义选项：品牌编辑器和托管 UI 品牌。您可以在 Amazon Cognito 控制台和用户池 API 操作中选择自己的首选体验。

品牌编辑器

品牌编辑器是最新的用户池用户界面体验（托管登录）的最新自定义选项。品牌编辑器是用于托管登录资产和样式的无代码可视化编辑器，以及一组 API 操作，用于对大量配置选项进行编程配置。您使用域名和托管登录配置的用户池会自动呈现登录页面的品牌设计师版本。

托管用户界面（经典）品牌

托管用户界面（经典）品牌体验有两个选项：使用一组固定的样式选项修改层叠样式表 (CSS) 文件，以及添加自定义徽标图片。您可以在 Amazon Cognito 控制台中或通过设置 UICustomization API 操作来设置这些选项。在该服务推出时，Amazon Cognito只有这个选项。您使用域名和托管用户界面品牌版本配置的用户池会自动呈现登录页面的经典版本。您的功能计划也可能仅支持托管用户界面。

> **Note**
>
> 品牌编辑器和经典品牌体验会修改托管身份验证服务的视觉属性。目前，您无法修改托管登录页面上显示的文本，只能将本地化应用于几种语言中的一种。有关本地化的更多信息，请参阅托管登录本地化。

选择品牌体验并分配风格

在 Amazon Cognito 控制台中，新用户池默认为托管登录品牌体验。您在托管登录可用之前设置的用户池将具有托管用户界面（经典）品牌。您可以在托管登录和托管用户界面品牌之间切换。当您更改品牌版本时，Amazon Cognito 会立即将更改应用于您的用户池域的用户互动页面。借助托管登录和托管用户界面，您的用户池可以为每个应用程序客户端设置一种样式。

每个应用程序客户端可以有不同的品牌风格，但是用户池域可以提供托管登录或托管用户界面。样式是应用于应用程序客户端的一组自定义设置。您可以为每个用户池设置一个自定义域和一个前缀域。您可以为自定义域名和前缀域名分配不同的品牌版本。但是，如果您还有自定义域，则前缀域无法完全正常运行，`.well-known`OIDC 发现端点仅提供自定义域路径。在具有此配置的用户池中，您只能将前缀域用于不需要端点发现 (`openid-configuration`) 的操作。由于用户池的这些属性，您可以有效地为每个用户池选择一个品牌版本。

在将域设置为托管登录品牌版本的用户池中，您可以为应用程序客户端分配样式。样式是一组视觉设置，由图像文件、显示选项、CSS 值组成。当您为应用程序客户端分配样式时，Amazon Cognito 会立即将您的更新推送到您的用户交互式登录页面。Amazon Cognito 使用您选择的品牌版本和您对其应用的自定义来呈现您的用户交互式页面。

更新和删除样式

创建样式时，将其链接到应用程序客户端。要更改应用程序客户端的样式分配，必须先删除原始样式。目前，您无法在样式之间复制设置。您必须以编程方式执行此操作。要在样式和应用程序客户端之间复制设置，请DescribeManagedLoginBranding通过 API 操作获取样式的设置，然后使用CreateManagedLoginBranding或应用这些设置UpdateManagedLoginBranding。您无法更改应用程序客户端的指定样式，只能删除原始样式并设置新的样式。有关使用 API 和 SDK 操作管理样式的更多信息，请参阅用于托管登录品牌的 API 和 SDK 操作。

> **Note**
>
> 创建或更新品牌风格的程序化请求的请求大小必须不超过 2 MB。如果您的请求大于此限制，请将您的请求分成多个`UpdateManagedLoginBranding`请求，请求的参数组不超过最大请求大小。这些请求不会导致将未指定的参数设置为默认值，因此您可以发送部分请求，而不会对现有设置产生任何影响。

您可以从托管登录菜单中删除 Amazon Cognito 控制台中的样式。在 “样式” 下，选择要删除的样式，然后选择 “删除样式”。

简而言之，为域名分配品牌的过程包括以下步骤。

1. 创建域名并设置品牌版本。
2. 创建品牌风格并将其分配给应用程序客户端。

为应用程序客户端分配样式

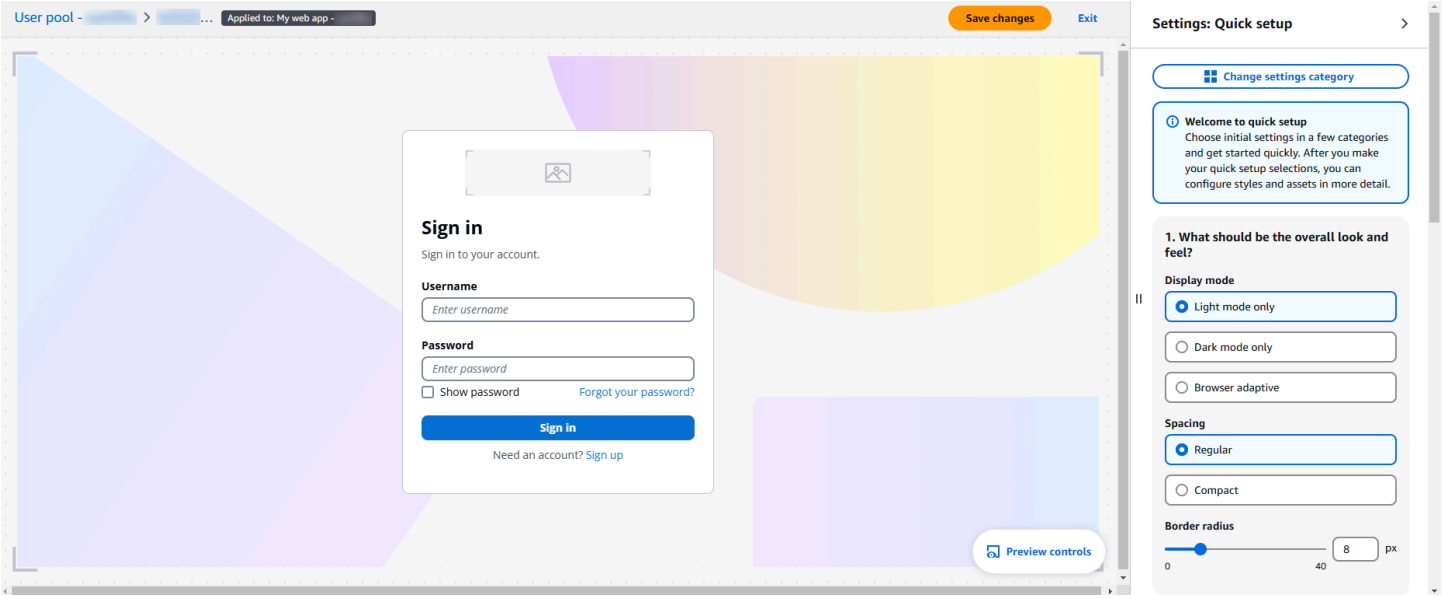
1. 在用户池的域名菜单中，创建一个域名并将品牌版本设置为托管登录。
2. 导航到托管登录菜单。在 “样式” 下，选择 “创建样式”。
3. 选择要为其分配样式的应用程序客户端，或者创建一个新的应用程序客户端。
4. 要开始配置品牌设置，请选择启动品牌编辑器。

主题

- 品牌编辑器和自定义托管登录
- 自定义托管 UI（经典）品牌

品牌编辑器和自定义托管登录

品牌编辑器是管理登录网页的可视化设计和编辑工具。它内置在亚马逊 Cognito 主机中。在品牌编辑器中，您可以从登录页面的预览开始，然后进入快速设置选项或带有高级选项的详细视图。您可以修改和预览样式参数或添加自定义背景图像和徽标。您可以配置浅色模式和深色模式。



首先，创建一种可以应用于用户池或应用程序客户端的样式。

要开始使用品牌编辑器

1. 通过 “域名” 选项卡创建域名，或更新现有域名。在品牌版本下，将您的域名设置为使用托管登录。
2. 删除现有的应用程序客户端样式（如果有）。
   a. 在应用程序客户端菜单中，选择您的应用程序客户端。
   b. 在托管登录方式下，选择分配给您的应用程序客户端的样式。
   c. 选择 “删除样式”。确认您的选择。
3. 导航到用户池中的托管登录菜单。如果您还没有，请按照提示选择包含托管登录的功能计划。如果您想在不进行更改的情况下查看品牌编辑器，也可以选择 “预览此功能”。
4. 在 “样式” 下，选择 “创建样式”。
5. 选择要为其分配样式的应用程序客户端，然后选择创建。您也可以创建新的应用程序客户端。
6. 亚马逊 Cognito 控制台启动品牌编辑器。
7. 选择要开始编辑的选项卡，或者选择启动编辑器并进入快速设置。以下选项卡可用：

预览

在托管登录页面中查看您当前选择的内容。

基础

设置总体主题，配置指向外部身份提供商的链接，并设置表单字段的样式。

组件

为页眉、页脚和各个 UI 元素配置样式。

8. 要选择初始设置，请进入快速设置。选择 “更改设置” 类别，然后选择 “快速设置”。选择 “继续” 后，品牌编辑器将启动，其中包含一组基本选项供您配置。

文本和本地化

您无法在品牌编辑器中修改或本地化文本。相反，在您分发给用户的 URL 中添加`lang`查询参数。此参数会使您的托管登录页面本地化为几种可用语言之一。有关更多信息，请参阅 托管登录本地化。

快速设置

Launch 品牌编辑器按钮为您的托管登录配置加载可视化编辑器，您可以在其中从各种主要的自定义选项中进行选择。当您做出选择时，Amazon Cognito 会在预览窗口中呈现您的托管登录更改。要返回详细设置菜单，请选择更改设置类别按钮。

整体外观和感觉应该如何？

为托管登录配置基本主题设置。

显示模式

为托管登录选择浅模式、暗模式或自适应体验。当 Amazon Cognito 呈现托管登录时，自适应设置会遵循用户的浏览器偏好。选择浏览器自适应模式时，可以为浅色和深色模式选择不同的颜色和徽标图像。

Spacing

设置页面中元素之间的默认间距。

边框半径

设置元素外边框的圆角深度。

页面背景应该是什么样子？

背景类型

“显示图像” 复选框指示您是想要背景图像还是要设置纯色背景色。

1. 要使用图像，请选择 “显示图像”，然后为浅色和深色模式选择背景图像。您还可以为图像未覆盖的背景区域设置深色模式和浅色模式的页面背景颜色。
2. 要仅使用背景颜色，请取消选择 “显示图像”，然后选择浅色模式和深色模式页面背景颜色。

表格应该是什么样子？

为托管登录的表单元素配置设置。表单项的示例包括登录和代码提示。

水平对齐

设置表单域的水平对齐方式。

表单徽标

设置徽标图片的位置。

徽标图片

选择要包含在浅色和深色模式的表单元素中的徽标图像文件。要上传图片，请选择徽标图片下拉列表，选择添加新资源，然后添加徽标文件。

主要品牌颜色

为浅色和深色模式设置主题颜色。此颜色将作为背景色应用于所有归类为主要元素的元素。

标题应该是什么样子？

选择是否要在托管登录页面中添加标题。标题可以包含徽标图片。

标题徽标

设置标题中徽标图片的位置。

徽标图片

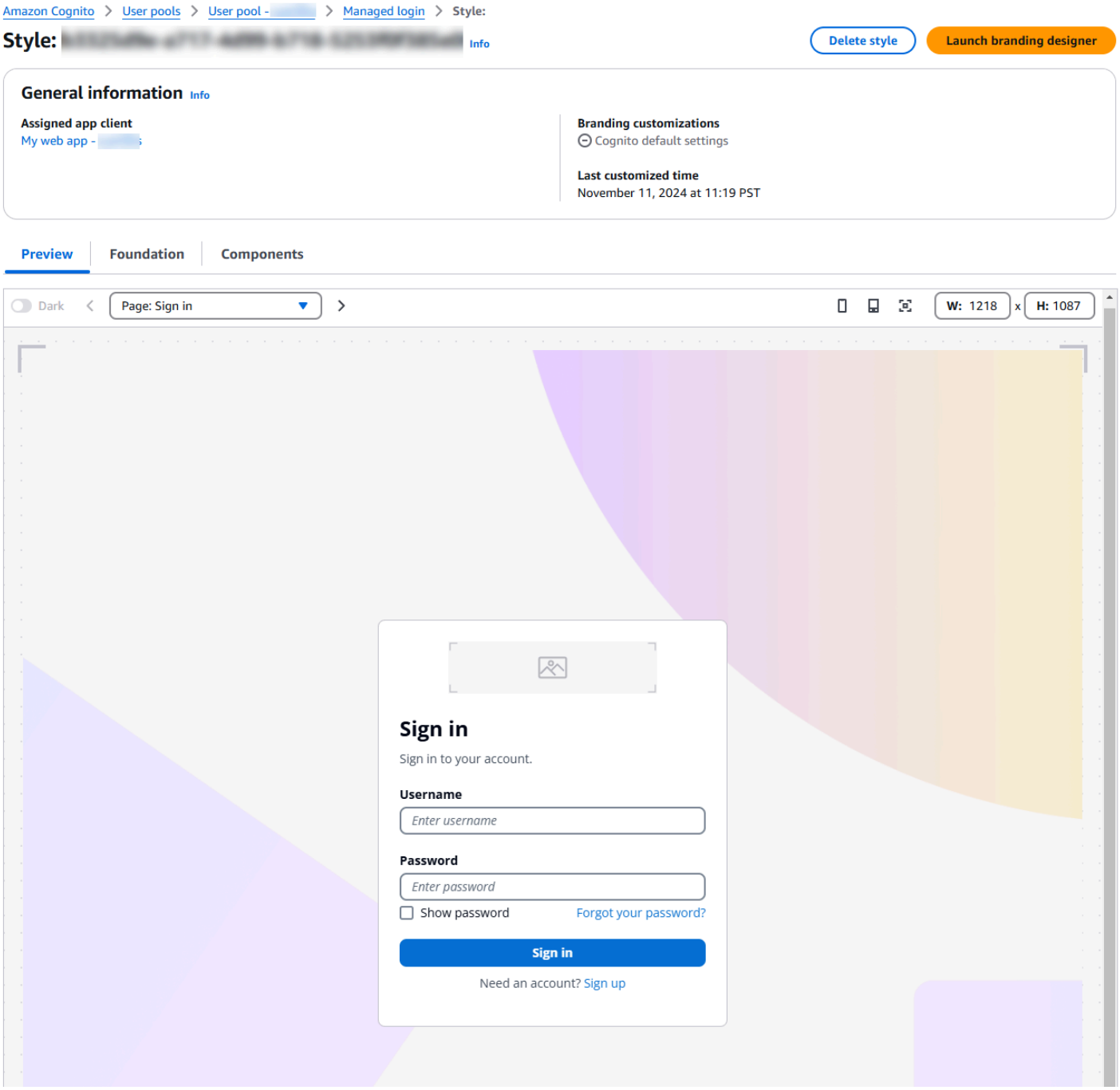
选择要包含在标题中的徽标位置和徽标图像文件。要上传图片，请选择徽标图片下拉列表，选择添加新资源，然后添加徽标文件。

标头背景颜色

设置标题背景的浅色和深色模式颜色。

详细设置

在详细设置视图中，您可以修改基础和组件中的各个组件。“预览”选项卡显示当前上下文中包含您的自定义设置的托管登录的预览。



要进入组件的可视化编辑器，请在组件的磁贴中选择编辑图标。在主题工作室编辑器中，您可以使用更改设置类别按钮在组件之间切换。

基础

应用程序风格

配置托管登录配置的基础知识。此类别包含整体主题、文本间距以及页眉和页脚的设置。

显示模式

为托管登录页面选择浅模式、暗模式或自适应体验。选择浏览器自适应模式时，可以为浅色和深色模式选择不同的颜色和徽标图像。

Spacing

设置页面中元素之间的默认间距。

身份验证行为

为将您的用户连接到外部身份提供商的按钮配置样式（IdPs）。本节包括域搜索输入选项，让托管登录提示用户输入电子邮件地址，并将其与其 SAML 身份提供商标识符进行匹配。

表单行为

配置输入表单的样式：输入的位置、颜色和元素的对齐方式。

组件

按钮

Amazon Cognito 在托管登录页面上呈现的按钮样式。

分隔线

托管登录元素（例如输入表单和外部提供商登录选择器）之间的分隔线和边界的样式。

下拉菜单

下拉菜单的样式。

网页图标

Amazon Cognito 为标签和书签图标提供的图片样式。

对焦环

用于表示当前选定输入的高亮显示样式。

表单容器

绑定表单的元素的样式。

全局页脚

Amazon Cognito 在托管登录页面底部显示的页脚样式。

全局标题

Amazon Cognito 在托管登录页面顶部显示的标题样式。

指示

错误和成功消息的样式。

选项控件

复选框、多选和其他输入提示的样式。

页面背景

托管登录整体背景的样式。

输入

表单字段输入提示的样式。

链接

托管登录页面中超链接的样式。

页面文本

页内文本的样式。

字段文本

表单输入周围的文本样式。

用于托管登录品牌的 API 和 SDK 操作

您还可以使用 API 操作CreateManagedLoginBranding和将品牌应用于托管登录风格UpdateManagedLoginBranding。这些操作非常适合为其他应用程序客户端或用户池创建相同或稍作修改的品牌样式版本。使用 API 操作查询现有样式的托管登录品牌DescribeManagedLoginBranding，然后根据需要修改输出并将其应用于其他资源。

该`UpdateManagedLoginBranding`操作不会更改您的样式所应用的应用程序客户端。它仅更新分配给应用程序客户端的现有样式。要完全替换应用程序客户端的样式，请删除现有样式，DeleteManagedLoginBranding然后使用指定新样式`CreateManagedLoginBranding`。在 Amazon Cognito 控制台中，情况也是如此：您必须删除现有样式并创建新样式。

在 API 或 SDK 请求中设置托管登录品牌需要将您的设置嵌入到转换为`Document`数据类型的 JSON 文件中。以下是您可以添加的图片以及生成配置品牌风格的编程请求的指南。

图像资产

CreateManagedLoginBranding并UpdateManagedLoginBranding包括一个`Assets`参数。此参数是一组采用 base64 编码的二进制格式的图像文件。

> **Note**
>
> 创建或更新品牌风格的程序化请求的请求大小必须不超过 2 MB。您请求中的资产可能会使其超过此限制。如果是这种情况，请将您的请求分解为多个`UpdateManagedLoginBranding`请求，以获取不超过最大请求大小的参数组。这些请求不会导致将未指定的参数设置为默认值，因此您可以发送部分请求，而不会对现有设置产生任何影响。

有些资源对您可以提交的文件类型有限制。

资产	接受的文件扩展名
FAVICON_ICO	ico
FAVICON_SVG	svg
电子邮件_图片	png、svg、jpeg

资产	接受的文件扩展名
短信_图片	png、svg、jpeg
AUTH_APP_GRAPH	png、svg、jpeg
密码_图形	png、svg、jpeg
密码_图形	png、svg、jpeg
PAGE_HEADER_LOGO	png、svg、jpeg
页面标题_背景	png、svg、jpeg
PAGE_FOOTER_LOGO	png、svg、jpeg
页面_页脚_背景	png、svg、jpeg
页面_背景	png、svg、jpeg
表单背景	png、svg、jpeg
表单徽标	png、svg、jpeg
IDP_BUTTON_ICON	ico，svg

SVG 类型的文件支持以下属性和元素。

Attributes

```
accent-height, accumulate, additivive, alignment-baseline, ascent, attributename,
 attributetype, azimuth, basefrequency, baseline-shift, begin, bias, by, class,
 clip, clip-path, clip-rule, color, color-interpolation, color-interpolation-
filters, color-profile, color-rendering, cx, cy, d, dx, dy, diffuseconstant,
 direction, display, divisor, dur, edgemode, elevation, end, fill, fill-opacity,
 fill-rule, filter, filterunits, flood-color, flood-opacity, font-family, font-
size, font-size-adjust, font-stretch, font-style, font-variant, font-weight, fx,
 fy, g1, g2, glyph-name, glyphref, gradientunits, gradienttransform, height, href,
 id, image-rendering, in, in2, k, k1, k2, k3, k4, kerning, keypoints, keysplines,
 keytimes, lang, lengthadjust, letter-spacing, kernelmatrix, kernelunitlength,
 lighting-color, local, marker-end, marker-mid, marker-start, markerheight,
 markerunits, markerwidth, maskcontentunits, maskunits, max, mask, media,
```

```
 method, mode, min, name, numoctaves, offset, operator, opacity, order, orient,
 orientation, origin, overflow, paint-order, path, pathlength, patterncontentunits,
 patterntransform, patternunits, points, preservealpha, preserveaspectratio, r,
 rx, ry, radius, refx, refy, repeatcount, repeatdur, restart, result, rotate,
 scale, seed, shape-rendering, specularconstant, specularexponent, spreadmethod,
 stddeviation, stitchtiles, stop-color, stop-opacity, stroke-dasharray, stroke-
dashoffset, stroke-linecap, stroke-linejoin, stroke-miterlimit, stroke-opacity,
 stroke, stroke-width, style, surfacescale, tabindex, targetx, targety, transform,
 text-anchor, text-decoration, text-rendering, textlength, type, u1, u2, unicode,
 values, viewbox, visibility, vert-adv-y, vert-origin-x, vert-origin-y, width, word-
spacing, wrap, writing-mode, xchannelselector, ychannelselector, x, x1, x2, xmlns,
 y, y1, y2, z, zoomandpan
```

Elements

```
svg, a, altglyph, altglyphdef, altglyphitem, animatecolor, animatemotion,
 animatetransform, audio, canvas, circle, clippath, defs, desc, ellipse, filter,
 font, g, glyph, glyphref, hkern, image, line, lineargradient, marker, mask,
 metadata, mpath, path, pattern, polygon, polyline, radialgradient, rect, stop,
 style, switch, symbol, text, textpath, title, tref, tspan, video, view, vkern,
 feBlend, feColorMatrix, feComponentTransfer, feComposite, feConvolveMatrix,
 feDiffuseLighting, feDisplacementMap, feDistantLight, feFlood, feFuncA, feFuncB,
 feFuncG, feFuncR, feGaussianBlur, feMerge, feMergeNode, feMorphology, feOffset,
 fePointLight, feSpecularLighting, feSpotLight, feTile, feTurbulence
```

用于托管登录品牌运营的工具

Amazon Cognito 管理托管登录品牌设置对象的 JSON 架构格式的文件。以下是如何以编程方式更新您的品牌风格。

更新用户池 API 中的品牌

1. 在 Amazon Cognito 控制台中，从用户池的托管登录菜单中创建默认的托管登录品牌样式。将其分配给应用程序客户端。
2. 例如，记录您为其创建样式的应用程序客户端的 ID `1example23456789`。
3. 使用设置为的 DescribeManagedLoginBrandingByClientAPI 请求检索品牌风格的`ReturnMergedResources`设置`true`。以下是一个示例请求正文。

```
{
   "ClientId": "1example23456789",
   "ReturnMergedResources": true,
```

```
    "UserPoolId": "us-east-1_EXAMPLE"
}
```

4. `DescribeManagedLoginBrandingByClient`使用您的自定义内容修改的输出。
   a. 响应正文被封装在不属于创建和更新操作语法的`ManagedLoginBranding`元素中。移除 JSON 对象的这个顶层。
   b. 要替换图像，请将该`Bytes`值替换为每个图像文件的 Base64 编码二进制数据。
   c. 要更新设置，请修改`Settings`对象的输出并将其包含在下一个请求中。Amazon Cognito 会忽略您的`Settings`对象中任何不在架构中但您在 API 响应中收到的值。
5. 在CreateManagedLoginBranding或UpdateManagedLoginBranding请求中使用更新的响应正文。如果此请求的大小超过 2 MB，请将其分成多个请求。除非您另行指定，否则这些操作适用于原始设置保持不变的`PATCH`模型。

自定义托管 UI（经典）品牌

您可以使用 Amazon Web Services Management Console、 Amazon CLI 或 API 为托管 UI 指定经典自定义设置。您可以上传要显示在应用程序中的自定义徽标图像。您还可以将一些层叠样式表 (CSS) 选项应用于用户界面的外观。

您可以自定义界面默认值，并使用特定设置覆盖各个应用程序客户端。Amazon Cognito 会将默认配置应用于每个没有客户端级设置的应用程序客户端。

在 Amazon Cognito 控制台和 API 请求中，设置用户界面自定义的请求大小不得超过 135 KB。在极少数情况下，请求标头、您的 CSS 文件和徽标的大小总和可能超过 135 KB。Amazon Cognito 对图像文件进行 Base64 编码。这会将 100KB 大小的图像增加到 130KB，保留 5KB 用于请求标头和 CSS。如果请求太大， Amazon Web Services Management Console 或您的 `SetUICustomization` API 请求将返回`request parameters too large`错误。请将您的徽标图像调整为不超过 100 KB，将 CSS 文件调整为不超过 3 KB。您无法单独设置 CSS 和徽标定制设置。

> **Note**
>
> 要自定义 UI，必须为用户群体设置域。

在经典品牌中指定自定义徽标

Amazon Cognito 将您的自定义徽标放在 登录端点 的输入字段上方居中。

为自定义托管 UI 徽标选择可缩放至 350 x 178 像素的 PNG、JPG 或 JPEG 文件。徽标文件的大小不得超过 100KB，或者在 Amazon Cognito 编码为 Base64 之后不超过 130KB。要设置输`ImageFile`入 SetUICustomization在 API 中，将您的文件转换为 Base64 编码的文本字符串，或者在中提供文件路径并让 Amazon Cognito 为您对其进行编码。 Amazon CLI

在经典品牌中指定 CSS 自定义设置

您可以为托管应用程序页面自定义 CSS，但存在以下限制：

- 您可以使用以下任意 CSS 类名：
  - `background-customizable`
  - `banner-customizable`
  - `errorMessage-customizable`
  - `idpButton-customizable`
  - `idpButton-customizable:hover`
  - `idpDescription-customizable`
  - `inputField-customizable`
  - `inputField-customizable:focus`
  - `label-customizable`
  - `legalText-customizable`
  - `logo-customizable`
  - `passwordCheck-valid-customizable`
  - `passwordCheck-notValid-customizable`
  - `redirect-customizable`
  - `socialButton-customizable`
  - `submitButton-customizable`
  - `submitButton-customizable:hover`
  - `textDescription-customizable`
- 属性值可以包含 HTML，但以下值除外：`@import`、`@supports`、`@page` 或者 `@media` 语句以及 Javascript。

您可以自定义以下 CSS 属性。

标签

- font-weight 是 100 的倍数 (从 100 到 900)。
- color 是文本颜色。必须是合法的 CSS 颜色值。

输入字段

- width 是以占包含块大小的百分比形式表示的宽度。
- height 是输入字段的高度，以像素 (px) 为单位。
- color 是文本颜色。它可以是任何标准 CSS 颜色值。
- background-color 是输入字段的背景色。它可以是任何标准 CSS 颜色值。
- border 是标准 CSS 边框值，用于指定您的应用程序窗口边框的宽度、透明度和颜色。宽度可以是从 1px 到 100px 的任何值。透明度可以是完全透明或不透明。颜色可以是任何标准颜色值。

文本描述

- padding-top 是文本描述上方的填充量。
- padding-bottom 是文本描述下方的填充量。
- display 可以是 `block` 或 `inline`。
- font-size 是文本描述的字体大小。
- color 是文本颜色。必须是合法的 CSS 颜色值。

提交按钮

- font-size 是按钮文本的字体大小。
- font-weight 是按钮文本的字体粗细：`bold`、`italic` 或 `normal`。
- 页边距是一个由四个值组成的字符串，用于指示按钮的上边距、右边距、下边距和左边距大小。
- font-size 是文本描述的字体大小。
- width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
- height 是按钮的高度，以像素 (px) 为单位。
- color 是按钮文本颜色。它可以是任何标准 CSS 颜色值。
- background-color 是按钮的背景色。它可以是任何标准颜色值。

横幅

- 填充是一个由四个值组成的字符串，用于指示横幅的上边距、右边距、下边距和左边距大小。
- background-color 是横幅的背景色。它可以是任何标准 CSS 颜色值。

提交按钮悬停

- color 是您将鼠标指针悬停在按钮上方时按钮的前景色。它可以是任何标准 CSS 颜色值。

- background-color 是您将鼠标指针悬停在按钮上方时按钮的背景色。它可以是任何标准 CSS 颜色值。

身份提供商按钮悬停

- color 是您将鼠标指针悬停在按钮上方时按钮的前景色。它可以是任何标准 CSS 颜色值。
- background-color 是您将鼠标指针悬停在按钮上方时按钮的背景色。它可以是任何标准 CSS 颜色值。

密码校验无效

- color 是 `"Password check not valid"` 消息的文本颜色。它可以是任何标准 CSS 颜色值。

背景

- background-color 是应用程序窗口的背景色。它可以是任何标准 CSS 颜色值。

错误消息

- 页边距是一个由四个值组成的字符串，用于指示上边距、右边距、下边距和左边距大小。
- padding 是边距大小。
- font-size 是字体大小。
- width 是错误消息的宽度，以占包含块大小的百分比形式表示。
- background 是错误消息的背景色。它可以是任何标准 CSS 颜色值。
- 边框是一个由三个值组成的字符串，用于指定边框的宽度、透明度和颜色。
- color 是错误消息文本颜色。它可以是任何标准 CSS 颜色值。
- box-sizing 用于向浏览器指示应包含的大小属性 (宽度和高度)。

身份提供商按钮

- height 是按钮的高度，以像素 (px) 为单位。
- width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
- text-align 是文本对齐设置。它可以是 `left`、`right` 或 `center`。
- margin-bottom 是下边距设置。
- color 是按钮文本颜色。它可以是任何标准 CSS 颜色值。
- background-color 是按钮的背景色。它可以是任何标准 CSS 颜色值。
- border-color 是按钮边框的颜色。它可以是任何标准 CSS 颜色值。

身份提供商描述

- padding-top 是描述上方的填充量。
- padding-bottom 是描述下方的填充量。

- display 可以是 `block` 或 `inline`。
- font-size 是描述的字体大小。
- co lor 是 IdP 部分标题的文本颜色，例如使用您的公司 ID 登录。必须是合法的 CSS 颜色值。

法律文本

- color 是文本颜色。它可以是任何标准 CSS 颜色值。
- font-size 是字体大小。

> **Note**
>
> 自定义 Legal text（法律文本）时，您自定义的是 We won't post to any of your accounts without asking first（如果未先询问，我们不会发布到您的任何账户）的消息，该消息显示在登录页面的社交身份提供商下方。

徽标

- max-width 是以占包含块大小的百分比形式表示的最大宽度。
- max-height 是以占包含块大小的百分比形式表示的最大高度。
- background-co lor 是带有透明部分的日志的背景颜色。必须是合法的 CSS 颜色值。

输入字段聚焦

- border-color 是输入字段的颜色。它可以是任何标准 CSS 颜色值。
- outline 是输入字段的边框宽度 (以像素为单位)。

社交按钮

- height 是按钮的高度，以像素 (px) 为单位。
- text-align 是文本对齐设置。它可以是 `left`、`right` 或 `center`。
- width 是按钮文本的宽度，以占包含块大小的百分比形式表示。
- margin-bottom 是下边距设置。

密码校验有效

- color 是 `"Password check valid"` 消息的文本颜色。它可以是任何标准 CSS 颜色值。

使用经典品牌自定义托管用户界面 Amazon Web Services Management Console

您可以使用为您的应用程序指定界面自定义设置。 Amazon Web Services Management Console

> **Note**
>
> 通过利用您的用户池的特定信息构建以下 URL 并将它键入到浏览器中，您可以查看具有自定义项的托管 UI： `https://<your_domain>/login?response_type=code&client_id=<your_app_client_id>&redirect_uri=<your_callbac`
> 在控制台中进行的更改出现之前，您可能必须等待长达 1 分钟才能刷新浏览器。
> 您的域显示在 App integration（应用程序集成）选项卡中，该选项卡位于 Domain（域）下方。您的应用程序客户端 ID 和回调 URL 在 App clients（应用程序客户端）下显示。

指定应用程序 UI 自定义设置

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 通过 “域名” 选项卡创建域名，或更新现有域名。在品牌推广版本下，将您的域名设置为使用托管用户界面（经典）。
4. 选择 “托管登录” 菜单。
5. 要为所有应用程序客户端自定义用户界面设置，请在 “托管用户界面设置” 下找到 “样式”，然后选择 “编辑”。
6. 要为一个应用程序客户端自定义界面设置，请转到应用程序客户端菜单并选择要修改的应用程序客户端，然后找到托管用户界面（经典）样式并选择覆盖。选择编辑。
7. 要上载自己的徽标图像文件，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）。
8. 要自定义托管 UI CSS，请下载 CSS template.css，然后使用您想要自定义的值修改模板。只有模板中包含的密钥才能与托管 UI 一起使用。添加的 CSS 密钥不会反映在您的 UI 中。自定义 CSS 文件后，请选择 Choose file（选择文件）或 Replace current file（替换当前文件）来上载您的自定义 CSS 文件。

使用用户池 API 中的经典品牌自定义托管用户界面，并使用 Amazon CLI

使用以下命令可为您的用户池指定应用程序 UI 自定义项。

使用以下 API 操作，获取用户群体的内置应用程序 UI 的 UI 定制设置。

- Amazon CLI: `aws cognito-idp get-ui-customization`
- Amazon API: GetUICustomization

使用以下 API 操作，设置用户群体的内置应用程序 UI 的 UI 定制设置。

- Amazon CLI 来自图像文件：`aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file fileb://"<path-to-logo-image-file>" --css ".label-customizable{ color: <color>;}"`
- Amazon CLI 图像编码为 Base64 二进制文本：`aws cognito-idp set-ui-customization --user-pool-id <your-user-pool-id> --client-id <your-app-client-id> --image-file <base64-encoded-image-file> --css ".label-customizable{ color: <color>;}"`
- Amazon API: SetUICustomization

使用 Lambda 触发器自定义用户池工作流

Amazon Cognito 使用 Amazon Lambda 函数来修改用户池的身份验证行为。您可以将您的用户群体配置为在用户首次注册之前、完成身份验证之后以及两者之间的几个阶段自动调用 Lambda 函数。您的函数可以修改身份验证流程的默认行为，发出 API 请求以修改您的用户池或其他 Amazon 资源，以及与外部系统通信。您的 Lambda 函数中的代码是您自己的。Amazon Cognito 会将事件数据发送到您的函数，等待函数处理数据，而且在大多数情况下，预计会出现一个响应事件，该事件反映您要对会话进行的任何更改。

在请求和响应事件系统中，您可以引入自己的身份验证挑战、在您的用户池与其他身份存储之间迁移用户、自定义消息以及修改 JSON Web 令牌（JWTs）。

Lambda 触发器可以自定义用户在您的用户群体中启动操作后 Amazon Cognito 向用户提供的响应。例如，您可以阻止原本会成功的用户登录。他们还可以对您的 Amazon 环境、外部环境 APIs、数据库或身份存储执行运行时操作。例如，迁移用户触发器可以将外部操作与 Amazon Cognito 中的更改相结合：您可以在外部目录中查找用户信息，然后根据该外部信息设置新用户的属性。

当您为用户群体分配 Lambda 触发器时，Amazon Cognito 会中断其原定设置流程，以从您的函数请求信息。Amazon Cognito 生成 JSON 事件并将其传递给您的函数。该事件包含有关您的用户旨在创建用户账户、登录、重置密码或更新属性的请求的信息。然后，您的函数有机会采取行动，或者将事件原封不动地发回。

下表总结了使用 Lambda 触发器自定义用户池操作的一些方法：

用户池流	操作	描述
自定义身份验证流程	定义身份验证质询	确定自定义身份验证流中的下一个挑战
	创建身份验证质询	在自定义身份验证流中创建挑战
	验证身份验证质询响应	确定响应在自定义身份验证流中是正确的
身份验证事件	the section called “身份验证前”	自定义验证以接受或拒绝登录请求
	the section called “身份验证后”	记录自定义分析的事件
	the section called “令牌生成前”	增加或隐藏令牌声明
注册	the section called “注册前”	执行接受或拒绝注册请求的自定义验证
	the section called “确认后”	为自定义分析添加自定义欢迎消息或事件日志记录
	the section called “迁移用户”	将用户从现有用户目录迁移到用户池
消息	the section called “自定义消息”	执行消息的高级自定义和本地化
令牌创建	the section called “令牌生成前”	在 ID 和访问令牌中添加或移除属性
电子邮件和 SMS 第三方提供商	the section called “自定义发件人”	使用第三方提供商发送 SMS 和电子邮件

主题

- 有关 Lambda 触发器的注意事项

- 添加用户池 Lambda 触发器
- 用户池 Lambda 触发器事件
- 用户池 Lambda 触发器通用参数
- 客户机元数据
- 将 API 操作连接到 Lambda 触发器
- 将 Lambda 触发器连接到用户群体功能操作
- 注册前 Lambda 触发器
- 确认后 Lambda 触发器
- 身份验证前 Lambda 触发器
- 身份验证后 Lambda 触发器
- 自定义身份验证质询 Lambda 触发器
- 令牌生成前 Lambda 触发器
- 迁移用户 Lambda 触发器
- 自定义消息 Lambda 触发器
- 自定义发件人 Lambda 触发器

有关 Lambda 触发器的注意事项

在为 Lambda 函数准备用户群体时，请考虑以下各项：

- Amazon Cognito 发送到 Lambda 触发器的事件可能会随着新功能推出而发生变化。响应和请求元素在 JSON 层次结构中的位置可能改变，或者可能会添加元素名称。在 Lambda 函数中，您预期会收到本指南中介绍的输入元素键值对，但是更严格的输入验证可能会导致您的函数失败。
- 您可以从 Amazon Cognito 发送到某些触发器的多个事件版本中选择一个。某些版本可能需要您接受对 Amazon Cognito 定价的更改。有关定价的更多信息，请参阅 Amazon Cognito 定价。要在中自定义访问令牌令牌生成前 Lambda 触发器，您必须使用精简版以外的功能计划配置用户池，并更新您的 Lambda 触发器配置以使用事件版本 2。
- Amazon Cognito 会同步调用 Lambda 函数，但 自定义发件人 Lambda 触发器 除外。Amazon Cognito 调用您的 Lambda 函数时，函数必须在 5 秒内响应。如果并非如此，并且可以重试调用，则 Amazon Cognito 会重试调用。3 次尝试失败后，该函数将超时。您无法更改此 5 秒钟超时值。有关更多信息，请参阅《 Amazon Lambda 开发人员指南》中的 Lambda 编程模型。

Amazon Cognito 不会重试返回调用错误且 HTTP 状态代码为 500-599 的函数调用。这些代码表示配置问题导致 Lambda 无法启动该函数。有关更多信息，请参阅中的错误处理和自动重试。Amazon Lambda

- 您无法在 Lambda 触发器配置中声明函数版本。默认情况下，Amazon Cognito 用户群体会调用您的函数的最新版本。但是，您可以将函数版本与别名相关联，并在CreateUserPool或UpdateUserPoolAPI 请求中`LambdaArn`将触发器设置为别名 ARN。此选项在 Amazon Web Services Management Console中不可用。要了解有关别名的更多信息，请参阅《Amazon Lambda 开发人员指南》中的 Lambda 函数别名。
- 如果您删除某个 Lambda 触发器，必须更新用户池中的相应触发器。例如，如果您删除身份验证后触发器，则必须在相应用户池中将 Post authentication（身份验证后）触发器设置为 none（无）。
- 如果您的 Lambda 函数没有向 Amazon Cognito 返回请求和响应参数，或者返回错误，则身份验证事件将无法成功。您可以在函数中返回错误，以阻止用户注册、身份验证、令牌生成或其身份验证流程中任何其他调用 Lambda 触发器的阶段。

  托管登录会返回 Lambda 触发器生成的错误作为登录提示上方的错误文本。Amazon Cognito 用户群体 API 以 `[trigger] failed with error [error text from response]` 格式返回触发器错误。最佳做法是，仅在 Lambda 函数中生成您希望用户看到的错误。使用输出方法`print()`，例如将任何敏感信息或调试信息 CloudWatch 记录到 Logs 中。有关示例，请参阅注册前示例：如果用户名少于五个字符，则拒绝注册。
- 您可以在另一个函数中添加 Lambda 函数 Amazon Web Services 账户 作为用户池的触发器。您必须使用CreateUserPool和 UpdateUserPoolAPI 操作添加跨账户触发器，或者在和中 Amazon CloudFormation 添加其等效项。 Amazon CLI您无法在中添加跨账户功能。 Amazon Web Services Management Console
- 当您在 Amazon Cognito 控制台中添加 Lambda 触发器时，Amazon Cognito 会向您的函数添加一个基于资源的策略，允许您的用户群体调用该函数。当您在 Amazon Cognito 控制台之外创建 Lambda 触发器（包括跨账户函数）时，您必须向 Lambda 函数的基于资源的策略添加权限。您添加的权限必须允许 Amazon Cognito 代表您的用户群体调用函数。您可以从 Lambda 控制台添加权限或使用 Lambd a API 操作。AddPermission

  Lambda 基于资源的策略示例

  以下 Lambda 基于资源的策略示例授予 Amazon Cognito 有限调用 Lambda 函数的能力。Amazon Cognito 只能在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能调用函数。

```
{
```

```
    "Version": "2012-10-17",
    "Id": "default",
    "Statement": [
        {
            "Sid": "lambda-allow-cognito",
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

添加用户池 Lambda 触发器

使用控制台添加用户池 Lambda 触发器

1. 使用 Lambda 控制台创建 Lambda 函数。有关 Lambda 函数的更多信息，请参阅《Amazon Lambda 开发人员指南》https://docs.amazonaws.cn/lambda/latest/dg/。
2. 转到 Amazon Cognito 控制台，然后选择 User Pools（用户池）。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “扩展” 菜单并找到 Lambda 触发器。
5. 选择 Add a Lambda trigger（添加 Lambda 触发器）。
6. 基于您希望自定义的身份验证阶段，选择 Lambda 触发器 Category（类别）。
7. 选择 “分配 Lambda 函数”，然后选择与您的用户池 Amazon Web Services 区域 相同的函数。

> **Note**
>
> 如果您的 Amazon Identity and Access Management (IAM) 证书有权更新 Lambda 函数，则 Amazon Cognito 会添加基于 Lambda 资源的策略。通过此政策，Amazon Cognito 可以调用您选择的函数。如果登录凭证没有足够的 IAM 权限，则必须单独更新基于资源的策略。有关更多信息，请参阅 the section called “需知信息”。

8. 选择保存更改。
9. 您可以在 Lambda 控制台 CloudWatch 中使用来记录您的 Lambda 函数。有关更多信息，请参阅访问 Lambda 的 CloudWatch 日志。

用户池 Lambda 触发器事件

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，Lambda 函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。如果您的函数未经修改就返回了输入事件，则 Amazon Cognito 会继续使用默认行为。下面显示了所有 Lambda 触发器输入事件的通用参数。有关特定于触发器的事件语法，请查看本指南部分中每个触发器的事件架构。

JSON

```
{
    "version": "string",
    "triggerSource": "string",
    "region": AWSRegion,
    "userPoolId": "string",
    "userName": "string",
    "callerContext":
        {
            "awsSdkVersion": "string",
            "clientId": "string"
        },
    "request":
        {
            "userAttributes": {
                "string": "string",
                ....
            }
        },
    "response": {}
```

```
}
```

用户池 Lambda 触发器通用参数

version

您的 Lambda 函数的版本号。

triggerSource

触发 Lambda 函数的事件的名称。有关每个 triggerSource 的说明，请参阅将 Lambda 触发器连接到用户群体功能操作。

区域

Amazon Web Services 区域 作为一个`AWSRegion`实例。

userPoolId

用户池的 ID。

userName

当前用户的用户名。

callerContext

有关请求和代码环境的元数据。它包含字段awsSdkVersion和客户端 ID。

awsSdkVersion

生成请求的 Amazon SDK 版本。

clientId

用户群体应用程序客户端的 ID。

request

您的用户的 API 请求的详细信息。它包括以下字段以及触发器特定的任何请求参数。例如，Amazon Cognito 发送到预身份验证触发器的事件也将包含一个 `userNotFound` 参数。当您的用户尝试使用未注册的用户名登录时，您可以处理此参数的值以执行自定义操作。

userAttributes

用户属性名称和值的一个或多个键值对，例如 `"email": "john@example.com"`。

响应

此参数在原始请求中不包含任何信息。Lambda 函数必须将整个事件返回给 Amazon Cognito，并将任何返回参数添加到 `response`。要查看您的函数可以包含哪些返回参数，请参阅要使用的触发器的文档。

客户机元数据

您可以在 API 操作和令牌端点请求中向 Lambda 触发器函数提交自定义参数。借助客户端元数据，您的应用程序可以收集有关请求来源环境的其他信息。当您将客户端元数据传递给您的 Lambda 函数时，它们可以处理其他数据，并将其用于记录或自定义身份验证流程。客户端元数据是您选择和设计的字符串对，采用 JSON 键值格式。

客户端元数据示例用例

- 在注册时将地理位置数据传递给注册前触发器，防止从不想要的位置登录。
- 将租户 ID 数据传递给自定义挑战触发器，并向来自不同业务部门的客户发出不同的质询。
- 将用户的令牌传递给代币生成前的触发器，并生成代表发出 M2M 请求的委托人的日志。有关示例请求，请参阅具有基本授权的客户凭证。

以下是向注册前触发器传递客户端元数据的示例。

SignUp request

以下是一个包含客户端元数据的示例SignUp请求，Amazon Cognito 会将其传递给注册前触发器。

```
POST HTTP/1.1
Host: cognito-idp.us-east-1.amazonaws.com
X-Amz-Date: 20230613T200059Z
Accept-Encoding: gzip, deflate, br
X-Amz-Target: AWSCognitoIdentityProviderService.SignUp
User-Agent: <UserAgentString>
Authorization: AWS4-HMAC-SHA256 Credential=<Credential>, SignedHeaders=<Headers>,
 Signature=<Signature>
Content-Length: <PayloadSizeBytes>

{
    "ClientId": "1example23456789",
    "Username": "mary_major",
    "Password": "<Password>",
```

```
    "SecretHash": "<Secret hash>",
    "ClientMetadata": {
        "IpAddress" : "192.0.2.252",
        "GeoLocation" : "Netherlands (Kingdom of the) [NL]"
    }
    "UserAttributes": [
        {
            "Name": "name",
            "Value": "Mary"
        },
        {
            "Name": "email",
            "Value": "mary_major@example.com"
        },
        {
            "Name": "phone_number",
            "Value": "+12065551212"
        }
    ],
}
```

Lambda trigger input event

该请求会为您的预注册功能生成以下请求正文。

```
{
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "region": "us-west-2",
    "request": {
        "clientMetadata": {
            "GeoLocation": "Netherlands (Kingdom of the) [NL]",
            "IpAddress": "192.0.2.252"
        },
        "userAttributes": {
            "email": "mary_major@example.com",
            "name": "Mary",
            "phone_number": "+12065551212"
        },
        "validationData": null
    },
    "response": {
```

```
        "autoConfirmUser": false,
        "autoVerifyEmail": false,
        "autoVerifyPhone": false
    },
    "triggerSource": "PreSignUp_SignUp",
    "userName": "mary_major2",
    "userPoolId": "us-west-2_EXAMPLE",
    "version": "1"
}
```

machine-to-machine(M2M) 客户端凭证的客户端元数据

您可以在 M2M 请求中传递客户端元数据。客户机元数据是来自用户或应用程序环境的其他信息，这些信息可能有助于实现的结果令牌生成前 Lambda 触发器。在使用用户委托人的身份验证操作中，您可以将客户端元数据传递给 RespondToAuthChallengeAPI 请求正文中的令牌生成前触发器。AdminRespondToAuthChallenge由于应用程序通过直接向 M2M 请求来执行生成访问令牌的流程令牌端点，因此它们的模型不同。在客户端凭证令牌请求的 POST 正文中，将带有客户端元数据对象 URL-encoded (`x-www-form-urlencoded`) 的`aws_client_metadata`参数传递给字符串。有关示例请求，请参阅具有基本授权的客户凭证。以下是传递键值对的示例参数。`{"environment": "dev", "language": "en-US"}`

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US
%22%7D
```

临时用户属性：**`validationData`**

一些身份验证操作也有一个`validationData`参数。与客户元数据一样，这是一个将外部信息传递给 Lambda 触发器的机会，Amazon Cognito 不会自动收集这些信息。验证数据字段旨在为您的 Lambda 函数提供注册和登录操作中的其他用户上下文。 SignUp并AdminCreateUser传递`validationData`到注册前的触发器。 InitiateAuth并像`ClientMetadata`在输入事件`validationData`中一样将 API 请求正文AdminInitiateAuth传递给预身份验证并迁移用户触发器。

要将 API 操作映射到它们可以向其传递客户端元数据的函数，请参阅后面的触发器源部分。

将 API 操作连接到 Lambda 触发器

以下部分介绍 Amazon Cognito 从您的用户群体中的活动调用的 Lambda 触发器。

当您的应用程序通过 Amazon Cognito 用户池 API、托管登录或用户池终端节点登录用户时，Amazon Cognito 会根据会话上下文调用您的 Lambda 函数。有关 Amazon Cognito 用户群体 API 和用户群体

端点的更多信息，请参阅了解 API、OIDC 和托管登录页面身份验证。以下各节中的表格描述了导致 Amazon Cognito 调用函数的事件，以及 Amazon Cognito 在请求中包含的 `triggerSource` 字符串。

主题

- Amazon Cognito API 中的 Lambda 触发器
- 在托管登录中会触发亚马逊 Cognito 本地用户的 Lambda
- 针对联合身份用户的 Lambda 触发器

Amazon Cognito API 中的 Lambda 触发器

下表描述了 Lambda 触发器的源字符串，当您的应用程序创建、登录或更新本地用户时，Amazon Cognito 可以调用这些触发器。

Amazon Cognito API 中的本地用户触发器来源

API 操作	Lambda 触发器	触发器源
AdminCreateUser	注册前	`PreSignUp_AdminCreateUser`
	令牌生成前	`TokenGeneration_NewPasswordChallenge`
	自定义消息	`CustomMessage_AdminCreateUser`
	自定义电子邮件发件人	`CustomEmailSender_AdminCreateUser`
	自定义 SMS 发送人	`CustomSMSSender_AdminCreateUser`
SignUp	注册前	`PreSignUp_SignUp`
	自定义消息	`CustomMessage_SignUp`
	自定义电子邮件发件人	`CustomEmailSender_SignUp`

API 操作	Lambda 触发器	触发器源
	自定义 SMS 发送人	`CustomSMSSender_SignUp`
ConfirmSignUp AdminConfirmSignUp	确认后	`PostConfirmation_ConfirmSignUp`
InitiateAuth AdminInitiateAuth	身份验证前	`PreAuthentication_Authentication`
	定义身份验证质询	`DefineAuthChallenge_Authentication`
	创建身份验证质询	`CreateAuthChallenge_Authentication`

	令牌生成前	`TokenGeneration_Authentication`
		
		`TokenGeneration_AuthenticateDevice`
		
		`TokenGeneration_RefreshTokens`

	迁移用户	`UserMigration_Authentication`
	自定义消息	`CustomMessage_Authentication`

	自定义电子邮件发件人	`CustomEmailSender_AccountTakeOverNotification`
		
		`CustomEmailSender_Authentication`

API 操作	Lambda 触发器	触发器源
	自定义 SMS 发送人	`CustomSMSSender_Authentication`
ForgotPassword	迁移用户	`UserMigration_ForgotPassword`
	自定义消息	`CustomMessage_ForgotPassword`
	自定义电子邮件发件人	`CustomEmailSender_ForgotPassword`
	自定义 SMS 发送人	`CustomSMSSender_ForgotPassword`
ConfirmForgotPassword	确认后	`PostConfirmation_ConfirmForgotPassword`
UpdateUserAttributes AdminUpdateUserAttributes	自定义消息	`CustomMessage_UpdateUserAttribute`
	自定义电子邮件发件人	`CustomEmailSender_UpdateUserAttribute`
	自定义 SMS 发送人	`CustomSMSSender_UpdateUserAttribute`
VerifyUserAttributes	自定义消息	`CustomMessage_VerifyUserAttribute`
	自定义电子邮件发件人	`CustomEmailSender_VerifyUserAttribute`
	自定义 SMS 发送人	`CustomSMSSender_VerifyUserAttribute`

API 操作	Lambda 触发器	触发器源
GetTokensFromRefreshToken	令牌生成前	TokenGeneration_Authentication

在托管登录中会触发亚马逊 Cognito 本地用户的 Lambda

下表描述了 Lambda 触发器的源字符串，当本地用户使用托管登录登录登录您的用户池时，Amazon Cognito 可以调用这些触发器。

托管登录中的本地用户触发源

托管登录 URI	Lambda 触发器	触发器源
/signup	注册前	PreSignUp_SignUp
	自定义消息	CustomMessage_SignUp
	自定义电子邮件发件人	CustomEmailSender_SignUp
	自定义 SMS 发送人	CustomSMSSender_SignUp
/confirmuser	确认后	PostConfirmation_ConfirmSignUp
/login	身份验证前	PreAuthentication_Authentication
	令牌生成前	TokenGeneration_Authentication TokenGeneration_AuthenticateDevice TokenGeneration_RefreshTokens

托管登录 URI	Lambda 触发器	触发器源
	迁移用户	UserMigration_Authentication
	自定义消息	CustomMessage_Authentication
	自定义电子邮件发件人	CustomEmailSender_AccountTakeOverNotification CustomEmailSender_Authentication
	自定义 SMS 发送人	CustomSMSSender_Authentication
/forgotpassword	迁移用户	UserMigration_ForgotPassword
	自定义消息	CustomMessage_ForgotPassword
	自定义电子邮件发件人	CustomEmailSender_ForgotPassword
	自定义 SMS 发送人	CustomSMSSender_ForgotPassword
/confirmforgotpassword	确认后	PostConfirmation_ConfirmForgotPassword

针对联合身份用户的 Lambda 触发器

您可以使用以下 Lambda 触发器，为使用联合身份提供商登录的用户自定义用户池工作流。

> **Note**
>
> 联邦用户可以使用托管登录进行登录，也可以向其生成请求，以静默方式将他们重定向到其身份提供商登录页面。对端点授权您无法使用 Amazon Cognito 用户群体 API 登录联合用户。

联合身份用户触发器源

登录事件	Lambda 触发器	触发器源
首次登录	注册前	`PreSignUp_ExternalProvider`
	确认后	`PostConfirmation_ConfirmSignUp`
	令牌生成前	`TokenGeneration_HostedAuth`
后续登录	身份验证前	`PreAuthentication_Authentication`
	身份验证后	`PostAuthentication_Authentication`
	令牌生成前	`TokenGeneration_HostedAuth`

联合身份登录不会在您的用户池中调用任何自定义身份验证质询 Lambda 触发器、迁移用户 Lambda 触发器、自定义消息 Lambda 触发器 或者自定义发件人 Lambda 触发器。

将 Lambda 触发器连接到用户群体功能操作

每个 Lambda 触发器都在您的用户群体中发挥功能作用。例如，触发器可以修改您的注册流程，或添加自定义身份验证质询。Amazon Cognito 发送到 Lambda 函数的事件可以反映构成该函数角色的多个操作之一。例如，当您的用户注册时以及当您创建用户时，Amazon Cognito 会调用预注册触发器。同一功能角色的每个不同案例都有其自身的 `triggerSource` 值。您的 Lambda 函数可以根据调用该函数的操作以不同的方式处理传入事件。

当事件对应于触发器源时，Amazon Cognito 还会调用所有分配的函数。例如，当用户登录到您分配了迁移用户和预身份验证触发器的用户群体时，他们会同时激活这两个触发器。

注册、确认和登录 (身份验证) 触发器

触发器	triggerSource 值	事件
注册前	`PreSignUp_SignUp`	注册前。
注册前	`PreSignUp_AdminCreateUser`	在管理员创建新用户时做好注册准备。
注册前	`PreSignUp_ExternalProvider`	适用于外部身份提供商的注册前。
确认后	`PostConfirmation_ConfirmSignUp`	注册后确认。
确认后	`PostConfirmation_ConfirmForgotPassword`	忘记密码后确认。
身份验证前	`PreAuthentication_Authentication`	身份验证前。
身份验证后	`PostAuthentication_Authentication`	身份验证后。

自定义身份验证质询触发器

触发器	triggerSource 值	事件
定义身份验证质询	`DefineAuthChallenge_Authentication`	定义身份验证质询。
创建身份验证质询	`CreateAuthChallenge_Authentication`	创建身份验证质询。

触发器	triggerSource 值	事件
验证身份验证质询	`VerifyAuthChallengeResponse_Authentication`	验证身份验证质询响应。

令牌生成前触发器

触发器	triggerSource 值	事件
令牌生成前	`TokenGeneration_HostedAuth`	Amazon Cognito 通过您的托管登录登录页面对用户进行身份验证。
令牌生成前	`TokenGeneration_Authentication`	用户身份验证或令牌刷新已完成。
令牌生成前	`TokenGeneration_NewPasswordChallenge`	管理员创建用户。当用户必须更改临时密码时，Amazon Cognito 调用此项。
令牌生成前	`TokenGeneration_AuthenticateDevice`	结束用户设备身份验证。
令牌生成前	`TokenGeneration_RefreshTokens`	用户尝试刷新身份和访问令牌时调用。

迁移用户触发器

触发器	triggerSource 值	事件
用户迁移	`UserMigration_Authentication`	用户登录时进行迁移。
用户迁移	`UserMigration_ForgotPassword`	忘记密码流程中的用户迁移。

自定义消息触发器

触发器	triggerSource 值	事件
自定义消息	`CustomMessage_SignUp`	用户在您的用户池中注册时的自定义消息。
自定义消息	`CustomMessage_AdminCreateUser`	当您创建用户作为管理员并且 Amazon Cognito 向他们发送临时密码时的自定义消息。
自定义消息	`CustomMessage_ResendCode`	现有用户请求新的确认代码时的自定义消息。
自定义消息	`CustomMessage_ForgotPassword`	用户请求重置密码时的自定义消息。
自定义消息	`CustomMessage_UpdateUserAttribute`	用户更改其电子邮件地址或电话号码并且 Amazon Cognito 发送验证代码时的自定义消息。
自定义消息	`CustomMessage_VerifyUserAttribute`	用户添加电子邮件地址或电话号码并且 Amazon Cognito 发送验证代码时的自定义消息。
自定义消息	`CustomMessage_Authentication`	配置了 SMS MFA 的用户登录时的自定义消息。

自定义发件人触发器

触发器	triggerSource 值	事件

自定义发件人	`CustomEmailSender_SignUp`
	
	`CustomSmsSender_SignUp`
		当用户在您的用户池中注册时。

触发器	triggerSource 值	事件

自定义发件人	`CustomEmailSender_AdminCreateUser`
	
	`CustomSmsSender_AdminCreateUser`
		当你以管理员身份创建用户时，Amazon Cognito 会向他们发送一个临时密码。

自定义发件人	`CustomEmailSender_ForgotPassword`
	
	`CustomSmsSender_ForgotPassword`
		当您的用户请求重置密码时。

自定义发件人	`CustomEmailSender_UpdateUserAttribute`
	
	`CustomSmsSender_UpdateUserAttribute`
		当用户更改其电子邮件地址或电话号码时，Amazon Cognito 会发送验证码。

自定义发件人	`CustomEmailSender_VerifyUserAttribute`
	
	`CustomSmsSender_VerifyUserAttribute`
		当用户添加电子邮件地址或电话号码时，Amazon Cognito 会发送验证码。

自定义发件人	`CustomEmailSender_Authentication`
	
	`CustomSmsSender_Authentication`
		配置了短信、电子邮件 MFA 或 OTP 的用户登录时。

自定义发件人	`CustomEmailSender_AccountTakeOverNotification`	当您的威胁防护设置对用户的登录尝试采取自动操作并且风险级别的操作包括通知时。

注册前 Lambda 触发器

您可能想要在具有自助注册选项的用户池中自定义注册流程。注册前触发器的一些常见用途包括对新用户进行自定义分析和记录、应用安全和治理标准，或者将第三方 IdP 的用户链接到整合的用户配置文件。您可能还有不需要进行验证和确认的可信用户。

在 Amazon Cognito 完成新本地用户或联合用户创建之前，它会立即激活注册前 Lambda 函数。在使用可信身份提供商进行自助注册SignUp或首次登录时，以及使用创建用户时，您的用户池会调用此触发器。AdminCreateUser在注册过程中，您可以使用此函数，利用自定义逻辑来分析登录事件，并修改或拒绝新用户。

主题

- 注册前 Lambda 触发器参数
- 注册前示例：从注册的域自动确认用户
- 注册前示例：自动确认和自动验证所有用户
- 注册前示例：如果用户名少于五个字符，则拒绝注册

注册前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
         },
        "clientMetadata": {
            "string": "string",
            . . .
         }
    },
```

```
    "response": {
        "autoConfirmUser": "boolean",
        "autoVerifyPhone": "boolean",
        "autoVerifyEmail": "boolean"
    }
}
```

注册前请求参数

userAttributes

表示用户属性的一个或多个名称/值对。属性名称是键。

validationData

一个或多个包含用户属性数据的键值对，您的应用程序在创建新用户的请求中将这些数据传递给 Amazon Cognito。在您AdminCreateUser或 SignUpAPI 请求的 ValidationData 参数中将此信息发送到您的 Lambda 函数。

Amazon Cognito 不会将你的 ValidationData 数据设置为你创建的用户的属性。 ValidationData 是您为注册前 Lambda 触发器而提供的临时用户信息。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为注册前触发器指定的 Lambda 函数。您可以使用以下 API 操作中的 ClientMetadata参数将此数据传递给您的 Lambda 函数：AdminCreateUser、AdminRespondToAuthChallengeForgotPassword、和。SignUp

注册前响应参数

在响应中，如果您想要自动确认用户，则您可以将 `autoConfirmUser` 设置为 `true`。您可以将 `autoVerifyEmail` 设置为 `true`，以自动验证用户的电子邮件。您可以将 `autoVerifyPhone` 设置为 `true`，以自动验证用户的电话号码。

> **Note**
>
> `AdminCreateUser` API 触发注册前 Lambda 函数时，Amazon Cognito 会忽略响应参数 `autoVerifyPhone`、`autoVerifyEmail` 和 `autoConfirmUser`。

autoConfirmUser

设置为 `true` 以自动确认用户，否则设置为 `false`。

autoVerifyEmail

设置为 `true` 可以设置为所注册用户已通过验证的电子邮件地址，否则为 `false`。如果 `autoVerifyEmail` 设置为 `true`，则 `email` 属性必须具有有效的非空值。否则将出现错误，用户将无法完成注册。

如果选择 `email` 属性作为别名，则在设置了 `autoVerifyEmail` 时将为用户的电子邮件地址创建别名。如果已存在具有该电子邮件地址的别名，则别名将移动到新用户，以前用户的电子邮件地址将标记为未验证。有关更多信息，请参阅 自定义登录属性。

autoVerifyPhone

设置为 `true` 可以设置为所注册用户已通过验证的电话号码，否则为 `false`。如果 `autoVerifyPhone` 设置为 `true`，则 `phone_number` 属性必须具有有效的非空值。否则将出现错误，用户将无法完成注册。

如果选择 `phone_number` 属性作为别名，则在设置了 `autoVerifyPhone` 时将为用户的电话号码创建别名。如果已存在具有该电话号码的别名，则别名将移动到新用户，以前用户的电话号码将标记为未验证。有关更多信息，请参阅 自定义登录属性。

注册前示例：从注册的域自动确认用户

这是示例 Lambda 触发器代码。在 Amazon Cognito 处理注册请求之前，会立即调用注册前触发器。它使用自定义属性 custom:domain 自动确认来自特定电子邮件域的新用户。任何不在自定义域中的新用户都将添加到用户池，但不会自动确认。

Node.js

```
export const handler = async (event, context, callback) => {
  // Set the user pool autoConfirmUser flag after validating the email domain
  event.response.autoConfirmUser = false;

  // Split the email address so we can compare domains
  var address = event.request.userAttributes.email.split("@");

  // This example uses a custom attribute "custom:domain"
  if (event.request.userAttributes.hasOwnProperty("custom:domain")) {
```

```
    if (event.request.userAttributes["custom:domain"] === address[1]) {
      event.response.autoConfirmUser = true;
    }
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # It sets the user pool autoConfirmUser flag after validating the email domain
    event['response']['autoConfirmUser'] = False

    # Split the email address so we can compare domains
    address = event['request']['userAttributes']['email'].split('@')

    # This example uses a custom attribute 'custom:domain'
    if 'custom:domain' in event['request']['userAttributes']:
        if event['request']['userAttributes']['custom:domain'] == address[1]:
            event['response']['autoConfirmUser'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "testuser@example.com",
            "custom:domain": "example.com"
        }
    },
    "response": {}
}
```

注册前示例：自动确认和自动验证所有用户

此示例确认所有用户并将用户的 email 和 phone_number 属性设置为“已验证”（如果该属性存在）。此外，如果启用了别名，当设置了自动验证时，将为 phone_number 和 email 创建别名。

> **Note**
>
> 如果已存在具有相同电话号码的别名，则别名将移动到新用户，以前用户的 phone_number 将标记为未验证。电子邮件地址也是如此。为了防止这种情况发生，您可以使用用户池 ListUsers API 来查看是否有现有用户已在使用新用户的电话号码或电子邮件地址作为别名。

Node.js

```
exports.handler = (event, context, callback) => {
  // Confirm the user
  event.response.autoConfirmUser = true;

  // Set the email as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("email")) {
    event.response.autoVerifyEmail = true;
  }

  // Set the phone number as verified if it is in the request
  if (event.request.userAttributes.hasOwnProperty("phone_number")) {
    event.response.autoVerifyPhone = true;
  }

  // Return to Amazon Cognito
  callback(null, event);
};
```

Python

```
def lambda_handler(event, context):
    # Confirm the user
    event['response']['autoConfirmUser'] = True

    # Set the email as verified if it is in the request
    if 'email' in event['request']['userAttributes']:
        event['response']['autoVerifyEmail'] = True
```

```
    # Set the phone number as verified if it is in the request
    if 'phone_number' in event['request']['userAttributes']:
        event['response']['autoVerifyPhone'] = True

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "request": {
    "userAttributes": {
      "email": "user@example.com",
      "phone_number": "+12065550100"
    }
  },
  "response": {}
}
```

注册前示例：如果用户名少于五个字符，则拒绝注册

此示例检查注册请求中用户名的长度。如果用户输入的名称长度少于五个字符，则该示例将返回错误。

Node.js

```
export const handler = (event, context, callback) => {
    // Impose a condition that the minimum length of the username is 5 is imposed on
 all user pools.
    if (event.userName.length < 5) {
        var error = new Error("Cannot register users with username less than the
 minimum length of 5");
        // Return error to Amazon Cognito
        callback(error, event);
    }
    // Return to Amazon Cognito
    callback(null, event);
```

```
};
```

Python

```
def lambda_handler(event, context):
    if len(event['userName']) < 5:
        raise Exception("Cannot register users with username less than the minimum
 length of 5")
    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "userName": "rroe",
  "response": {}
}
```

确认后 Lambda 触发器

Amazon Cognito 会在注册用户确认其用户账户后调用此触发器。在您的确认后 Lambda 函数中，您可以发送自定义消息或添加自定义 API 请求。例如，您可以查询外部系统并为用户填充其他属性。Amazon Cognito 仅对在您的用户群体中注册的用户调用此触发器，而不会针对您使用管理员凭证创建的用户账户调用此触发器。

请求包含已确认用户的当前属性。您的用户池在ConfirmSignUp、AdminConfirmSignUp和上调用您的帖子确认功能。ConfirmForgotPassword当用户在托管登录中确认注册或密码重置时，也会运行此触发器。

主题

- 确认后 Lambda 触发器参数
- 确认后示例

确认后 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
            "userAttributes": {
                "string": "string",
                . . .
            },
            "clientMetadata": {
             "string": "string",
             . . .
            }
        },
    "response": {}
}
```

确认后请求参数

userAttributes

表示用户属性的一个或多个键值对。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为确认后触发器指定的 Lambda 函数。您可以使用以下 API 操作中的 ClientMetadata 参数将此数据传递给您的 Lambda 函数：AdminConfirmSignUp、ConfirmForgotPasswordConfirmSignUp、和。SignUp

确认后响应参数

预计响应中没有其他返回信息。

确认后示例

此示例 Lambda 函数将使用 Amazon SES 向用户发送确认电子邮件。有关更多信息，请参阅 Amazon Simple Email Service 开发人员指南。

Node.js

```
// Import required AWS SDK clients and commands for Node.js. Note that this requires
// the `@aws-sdk/client-ses` module to be either bundled with this code or included
// as a Lambda layer.
import { SES, SendEmailCommand } from "@aws-sdk/client-ses";
const ses = new SES();

const handler = async (event) => {
  if (event.request.userAttributes.email) {
    await sendTheEmail(
      event.request.userAttributes.email,
      `Congratulations ${event.userName}, you have been confirmed.`,
    );
  }
  return event;
};

const sendTheEmail = async (to, body) => {
  const eParams = {
    Destination: {
      ToAddresses: [to],
    },
    Message: {
      Body: {
        Text: {
          Data: body,
        },
      },
      Subject: {
        Data: "Cognito Identity Provider registration completed",
      },
    },
    // Replace source_email with your SES validated email address
    Source: "<source_email>",
  };
  try {
    await ses.send(new SendEmailCommand(eParams));
  } catch (err) {
    console.log(err);
  }
};
```

```
export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "request": {
        "userAttributes": {
            "email": "user@example.com",
            "email_verified": true
        }
    },
    "response": {}
}
```

身份验证前 Lambda 触发器

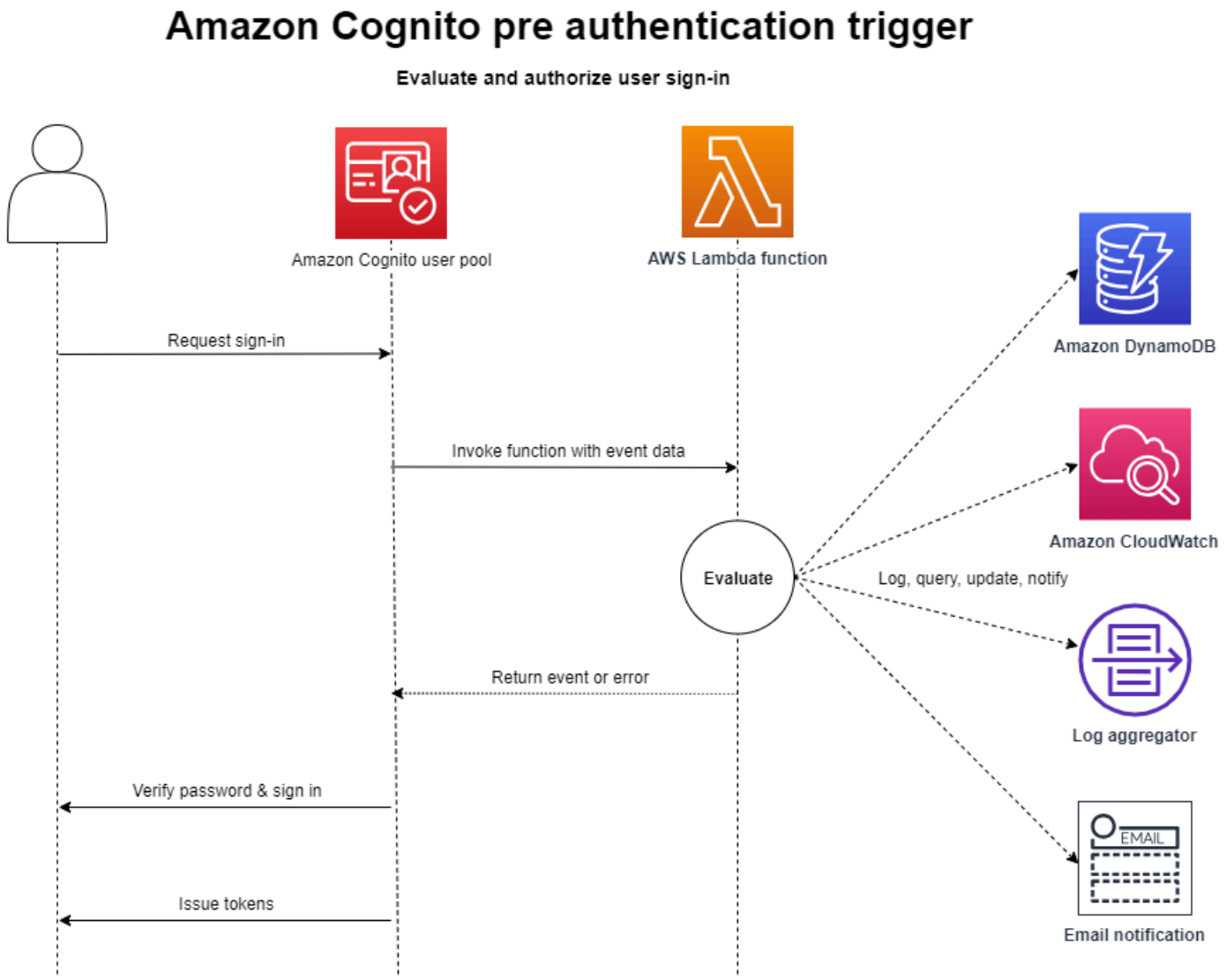
当用户尝试登录时，Amazon Cognito 会调用此触发器，以便您可以创建用于执行准备操作的自定义验证。例如，您可以拒绝身份验证请求或将会话数据记录到外部系统。

> **Note**
>
> 除非将用户池应用程序客户端的`PreventUserExistenceErrors`设置设置为`ENABLED`，否则当用户不存在时，此 Lambda 触发器不会激活。续订现有身份验证会话也不会激活此触发器。

主题

- 流程概述
- 身份验证前 Lambda 触发器参数
- 身份验证前示例

流程概述



该请求包含来自 `ClientMetadata` 值的客户端验证数据，该值由应用程序传递到用户群体 `InitiateAuth` 和 `AdminInitiateAuth` API 操作。

有关更多信息，请参阅 身份验证会话示例。

身份验证前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "validationData": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {}
}
```

身份验证前请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

当您将用户池客户端的 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充此布尔值。

validationData

一个或多个键/值对，包含用户的登录请求中的验证数据。要将此数据传递给您的 Lambda 函数，请使用InitiateAuth和 AdminInitiateAuthAPI 操作中的 ClientMetadata 参数。

身份验证前响应参数

Amazon Cognito 不会处理您的函数在响应中返回的任何附加信息。您的函数可以返回错误以拒绝登录尝试，或者使用 API 操作来查询和修改资源。

身份验证前示例

此示例函数阻止用户使用特定的应用程序客户端登录到您的用户群体。由于预身份验证 Lambda 函数不会在您的用户有现有会话时调用，因此，此函数仅阻止使用您想要屏蔽的应用程序客户端 ID 的新会话。

Node.js

```
const handler = async (event) => {
  if (
    event.callerContext.clientId === "user-pool-app-client-id-to-be-blocked"
  ) {
    throw new Error("Cannot authenticate users from this user pool app client");
  }

  return event;
};

export { handler };
```

Python

```
def lambda_handler(event, context):
    if event['callerContext']['clientId'] == "<user pool app client id to be
 blocked>":
        raise Exception("Cannot authenticate users from this user pool app client")

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "callerContext": {
        "clientId": "<user pool app client id to be blocked>"
    },
    "response": {}
```

```
}
```

身份验证后 Lambda 触发器

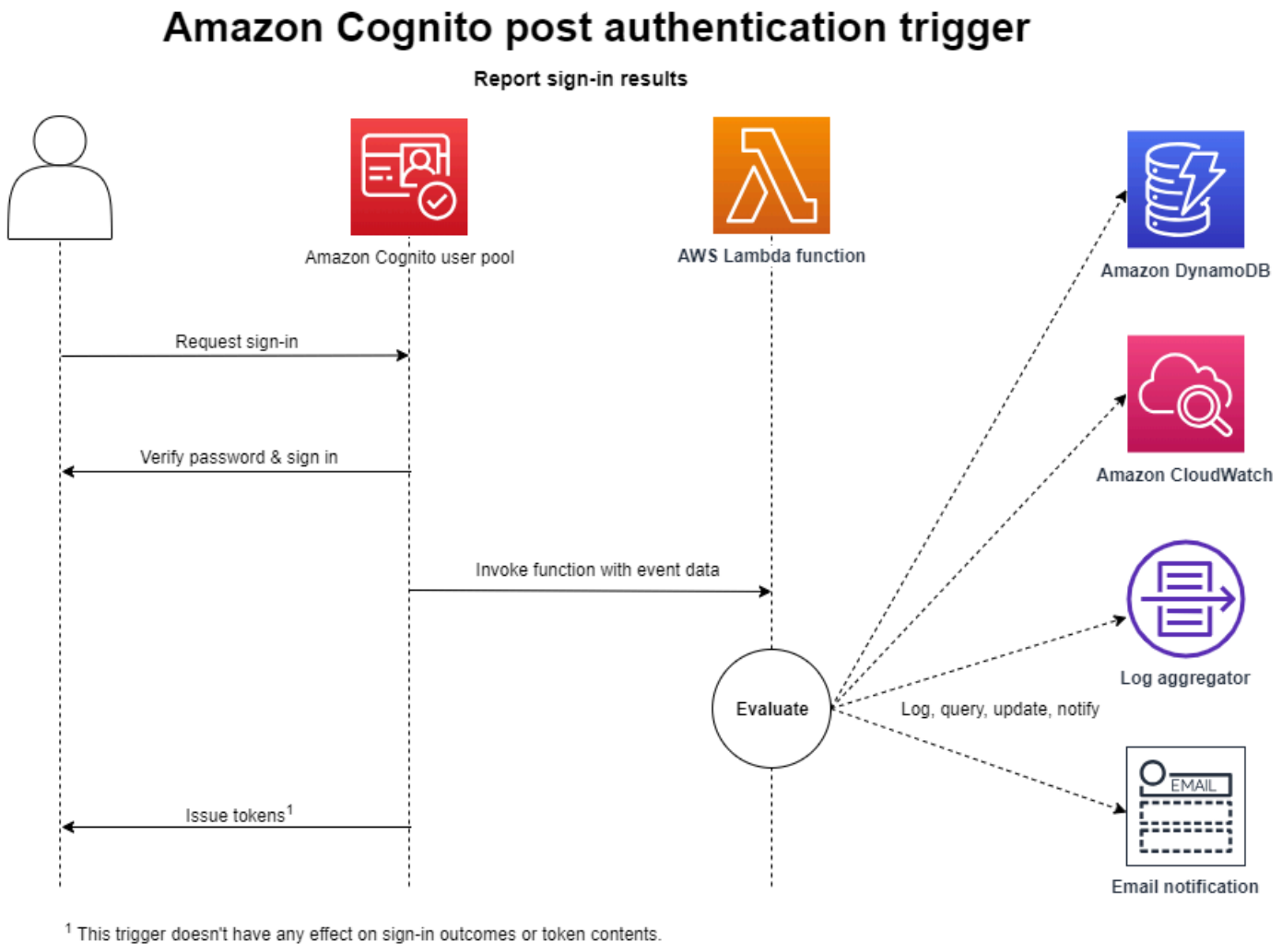
身份验证后触发器不会更改用户的身份验证流程。Amazon Cognito 会在身份验证完成后，在用户收到令牌之前调用此 Lambda。当您想要添加身份验证事件的自定义后处理时（例如，将在下次登录时反映的日志记录或用户配置文件调整），请添加身份验证后触发器。

不将请求正文返回给 Amazon Cognito 的身份验证后 Lambda 仍会导致身份验证无法完成。有关更多信息，请参阅 有关 Lambda 触发器的注意事项。

主题

- 身份验证流概述
- 身份验证后 Lambda 触发器参数
- 身份验证后示例

身份验证流概述



有关更多信息，请参阅 身份验证会话示例。

身份验证后 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
```

```
            . . .
        },
        "newDeviceUsed": boolean,
        "clientMetadata": {
            "string": "string",
            . . .
           }
       },
    "response": {}
}
```

身份验证后请求参数

newDeviceUsed

此标记指示用户是否已在新设备上登录。Amazon Cognito 仅在用户池的记住的设备值设置为 `Always` 或 `User Opt-In` 时设置此标记。

userAttributes

表示用户属性的一个或多个名称/值对。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为身份验证后触发器指定的 Lambda 函数。要将此数据传递给您的 Lambda 函数，您可以使用AdminRespondToAuthChallenge和RespondToAuthChallengeAPI 操作中的 ClientMetadata参数。Amazon Cognito 在传递给身份验证后函数的请求中不包含来自 ClientMetadata 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。

身份验证后响应参数

Amazon Cognito 不需要响应中任何额外的返回信息。您的函数可以使用 API 操作来查询和修改资源，或者将事件元数据记录到外部系统。

身份验证后示例

此身份验证后示例 Lambda 函数将成功登录后的数据发送到日志。 CloudWatch

Node.js

```
const handler = async (event) => {
  // Send post authentication data to Amazon CloudWatch logs
```

```
  console.log("Authentication successful");
  console.log("Trigger function =", event.triggerSource);
  console.log("User pool = ", event.userPoolId);
  console.log("App client ID = ", event.callerContext.clientId);
  console.log("User ID = ", event.userName);

  return event;
};

export { handler };
```

Python

```
import os
def lambda_handler(event, context):

    # Send post authentication data to Cloudwatch logs
    print ("Authentication successful")
    print ("Trigger function =", event['triggerSource'])
    print ("User pool = ", event['userPoolId'])
    print ("App client ID = ", event['callerContext']['clientId'])
    print ("User ID = ", event['userName'])

    # Return to Amazon Cognito
    return event
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "triggerSource": "testTrigger",
  "userPoolId": "testPool",
  "userName": "testName",
  "callerContext": {
      "clientId": "12345"
  },
  "response": {}
}
```

自定义身份验证质询 Lambda 触发器

在为 Amazon Cognito 用户池构建身份验证流程时，您可能会发现需要在内置流程的基础上对身份验证模型进行扩展。自定义质询触发器的一个常见使用场景是在用户名、密码和多重身份验证（MFA）之外实施额外的安全检查。自定义质询是您可以使用 Lambda 支持的编程语言生成的任何问题和回答。例如，在允许用户进行身份验证之前，您可能希望要求用户先破解验证码或回答安全问题。另一个潜在的需求是与专门的身份验证因素或设备集成。或者，您可能已经开发了使用硬件安全密钥或生物识别设备对用户进行身份验证的软件。自定义质询的身份验证成功的定义是，您的 Lambda 函数接受为正确的答案：例如，固定字符串或来自外部 API 的满意响应。

您可以使用自定义质询开始身份验证并完全控制身份验证过程，也可以在应用程序收到自定义质询之前执行用户名和密码身份验证。

自定义身份验证质询 Lambda 触发器：

定义

启动质询序列。确定您是要启动新的质询、将身份验证标记为已完成，还是要停止身份验证尝试。

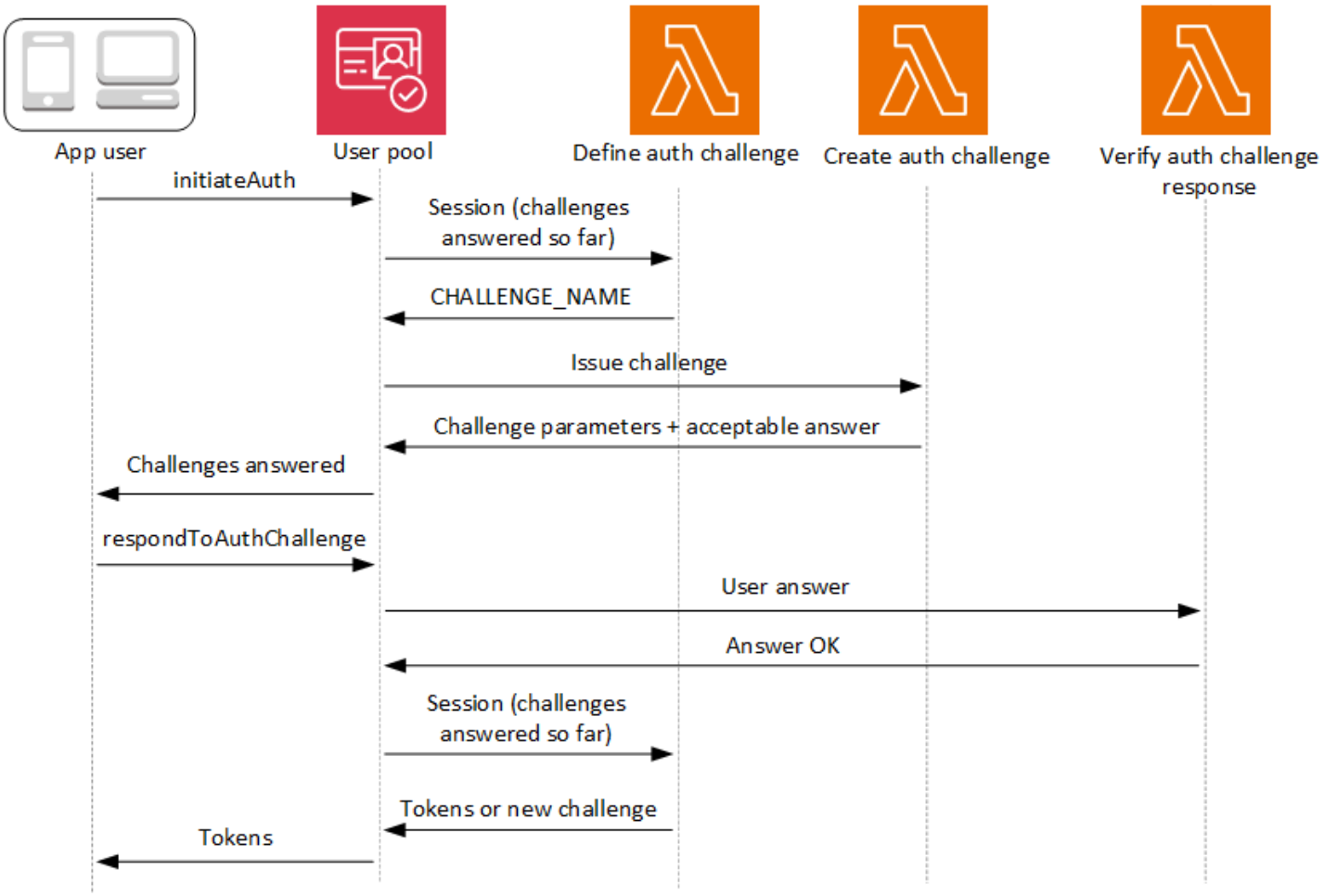
创建

向您的应用程序发出用户必须回答的问题。此函数可能会呈现安全问题或指向验证码的链接，您的应用程序应将其显示给用户。

验证

知道预期答案并将其与您的应用程序在质询响应中提供的答案进行比较。该函数可能会调用您的验证码服务的 API 来检索用户尝试的解决方案的预期结果。

这三个 Lambda 函数链接在一起，呈现出一种完全由您控制且由您自己设计的身份验证机制。由于自定义身份验证需要在您的客户端和 Lambda 函数中使用应用程序逻辑，因此您无法在托管登录中处理自定义身份验证。此身份验证系统需要开发人员付出额外的努力。您的应用程序必须使用用户池 API 执行身份验证流程，并使用自定义的登录界面来处理由此产生的挑战，该界面将问题呈现为自定义身份验证质询的中心。



有关实施自定义身份验证的更多信息，请参阅自定义身份验证流程和质询。

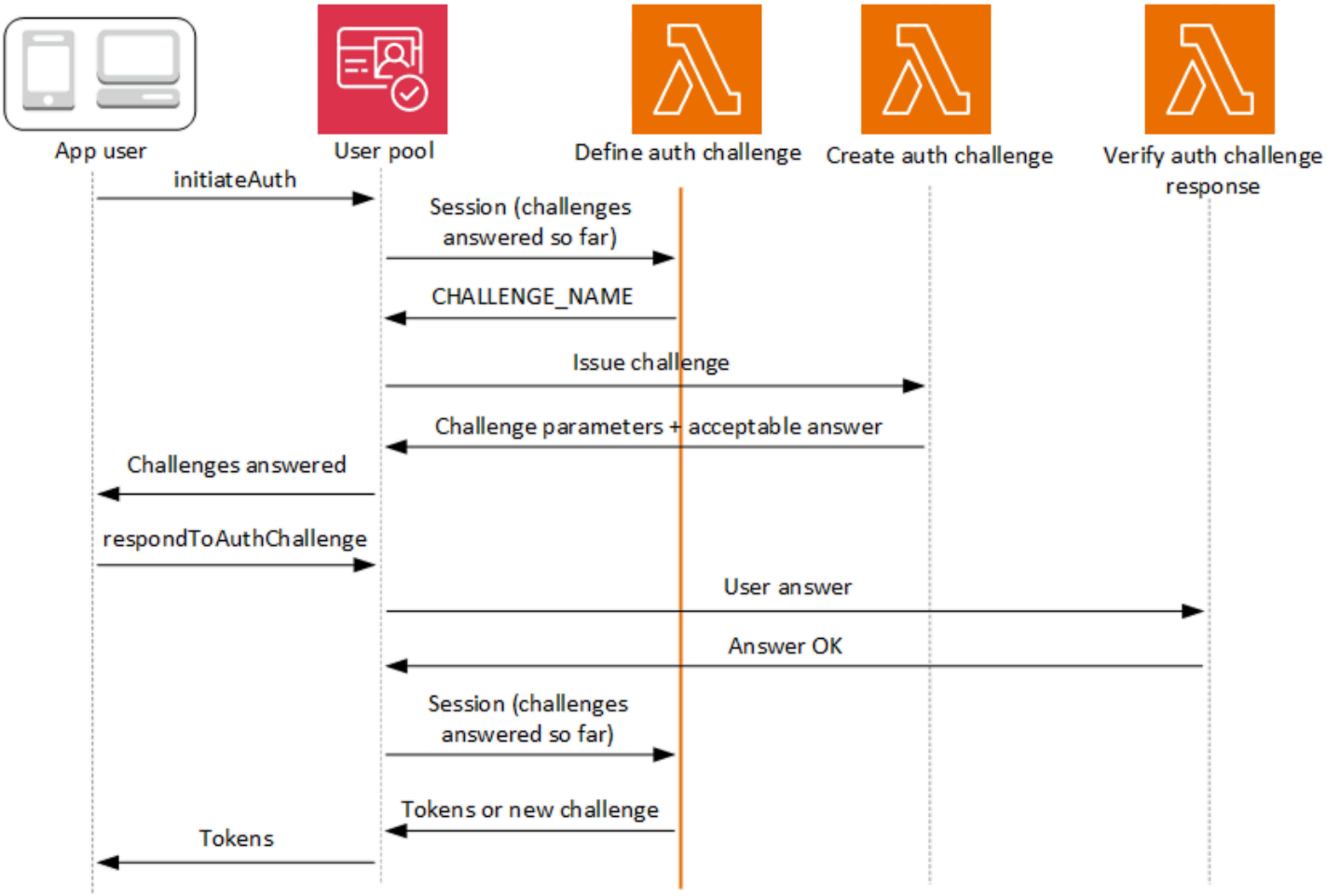
API 操作之间的身份验证InitiateAuth或AdminInitiateAuth、和RespondToAuthChallenge或AdminRespondToAuthChallenge。在此流程中，用户通过回答连续的质询进行身份验证，直到身份验证失败或用户获得令牌。质询回应可能是一个新的挑战。在这种情况下，您的应用程序会根据需要多次响应新的质询。当定义身份验证质询函数分析到目前为止的结果时，确定所有质询都已回答并返回 `IssueTokens` 时，身份验证就会成功。

主题

- 定义身份验证质询 Lambda 触发器
- 创建身份验证质询 Lambda 触发器
- 验证身份验证质询响应 Lambda 触发器

定义身份验证质询 Lambda 触发器

定义身份验证质询触发器是一个 Lambda 函数，用于在自定义身份验证流程中维护质询序列。它声明质询序列的成功或失败，并在序列尚未完成时设置下一个质询。



定义身份验证质询

Amazon Cognito 调用此触发器以启动自定义身份验证流程。

此 Lambda 触发器的请求包括 `session`。`session` 参数是一个数组，包含在当前身份验证流程中向用户显示的所有质询。请求还包含相应的结果。`session` 数组按照时间顺序存储质询详细信息 (`ChallengeResult`)。质询 `session[0]` 表示用户收到的第一个质询。

您可以让 Amazon Cognito 在发出自定义质询之前验证用户密码。当您在自定义质询流程中执行 SRP 身份验证时，请求频率限额身份验证类别中关联的任何 Lambda 触发器都将运行。过程概述如下：

1. 您的应用程序使用 `AuthParameters` 映射来调用 `InitiateAuth` 或 `AdminInitiateAuth`，以此来启动登录。参数必须包括 `CHALLENGE_NAME: SRP_A`，以及 `SRP_A` 和 `USERNAME` 的值。

2. Amazon Cognito 使用包含 `challengeName: SRP_A` 和 `challengeResult: true` 的初始会话，调用您定义的身份验证质询 Lambda 触发器。
3. 在收到这些输入后，您的 Lambda 函数发出 `challengeName: PASSWORD_VERIFIER`、`issueTokens: false`、`failAuthentication: false` 响应。
4. 如果密码验证成功，Amazon Cognito 会使用包含 `challengeName: PASSWORD_VERIFIER` 和 `challengeResult: true` 的新会话再次调用您的 Lambda 函数。
5. 为了启动您的自定义质询，Lambda 函数发出 `challengeName: CUSTOM_CHALLENGE`、`issueTokens: false` 和 `failAuthentication: false` 响应。如果您不想启动包含密码验证的自定义身份验证流程，可以使用 `AuthParameters` 映射（包括 `CHALLENGE_NAME: CUSTOM_CHALLENGE`）启动登录。
6. 质询循环将一直重复到所有质询得到应答。

以下在使用 SRP 流进行自定义身份验证之前的起始 `InitiateAuth` 请求的示例。

```
{
    "AuthFlow": "CUSTOM_AUTH",
    "ClientId": "1example23456789",
    "AuthParameters": {
        "CHALLENGE_NAME": "SRP_A",
        "USERNAME": "testuser",
        "SRP_A": "[SRP_A]",
        "SECRET_HASH": "[secret hash]"
    }
}
```

主题

- 定义身份验证质询 Lambda 触发器参数
- 定义身份验证质询示例

定义身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
```

```
    "request": {
        "userAttributes": {
            "string": "string",
                . . .
        },
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "challengeName": "string",
        "issueTokens": boolean,
        "failAuthentication": boolean
    }
}
```

定义身份验证质询请求参数

当 Amazon Cognito 调用您的 Lambda 函数时，Amazon Cognito 提供以下参数：

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

一个布尔值，当您的用户池客户端将 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充该值。值 `true` 表示用户 ID（用户名、电子邮件地址以及其他详细信息）不匹配任何现有用户。当 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，该服务不会向应用程序通知不存在的用户。我们建议您的 Lambda 函数保持相同的用户体验并考虑延迟。这样，不论用户是否存在，调用方都不会检测到不同的行为。

会话

`ChallengeResult` 元素的数组。每个数组包含以下元素：

challengeName

以下质询类型之一：CUSTOM_CHALLENGE、SRP_A、PASSWORD_VERIFIER、SMS_MFA、EMAIL_OTP、SOFTWARE_T
或 ADMIN_NO_SRP_AUTH。

当您的定义身份验证质询功能向已设置多重身份验证的用户发出 PASSWORD_VERIFIER 质询时，Amazon Cognito 会随后提出 SMS_MFA、EMAIL_OTP 或 SOFTWARE_TOKEN_MFA 质询。这些是多重身份验证代码的提示。在您的函数中，包括对来自 SMS_MFA、EMAIL_OTP 和 SOFTWARE_TOKEN_MFA 质询的输入事件的处理。您无需在定义身份验证质询函数中调用任何 MFA 质询。

> **Important**
>
> 在函数确定用户是否已成功通过身份验证以及是否应向其颁发令牌时，请始终检查定义身份验证质询函数中的 challengeName 以及它是否与预期值匹配。

challengeResult

如果用户成功完成质询，则设置为 true，否则设置为 false。

challengeMetadata

您的自定义质询的名称。仅当 challengeName 为 CUSTOM_CHALLENGE 时使用。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为定义身份验证质询触发器指定的 Lambda 函数。要将此数据传递给您的 Lambda 函数，您可以在AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 操作中使用ClientMetadata参数。调用 define auth 质询函数的请求不包括在 API 操作中的 ClientMetadata 参数中AdminInitiateAuth传递的数据。InitiateAuth

定义身份验证质询响应参数

在响应中，您可以返回身份验证流程的下一阶段。

challengeName

一个字符串，其中包含下一质询的名称。如果您希望向您的用户显示新的质询，请在此处指定质询名称。

issueTokens

如果您确定用户已充分完成了身份验证质询，则设置为 `true`。如果用户没有充分满足质询条件，则设置为 `false`。

failAuthentication

如果您想要终止当前的身份验证流程，则设置为 `true`。要继续当前的身份验证流程，请设置为 `false`。

定义身份验证质询示例

此示例针对身份验证定义一系列质询，并仅在用户成功完成所有质询后发布令牌。当用户使用`SRP_A`和`PASSWORD_VERIFIER`质询完成 SRP 身份验证时，此函数会向他们传递一个`CUSTOM_CHALLENGE`调用创建身份验证质询触发器的。结合我们的 c reate auth 挑战示例，这个序列为挑战三提供了一个验证码挑战，为挑战四提供了一个安全问题。

用户解决验证码并回答安全问题后，此功能确认您的用户池可以发行令牌。不需要 SRP 身份验证；您也可以将验证码和安全问题设置为挑战一和二。如果您的 define auth 质询函数未声明 SRP 挑战，则用户的成功完全取决于他们对您的自定义提示的响应。

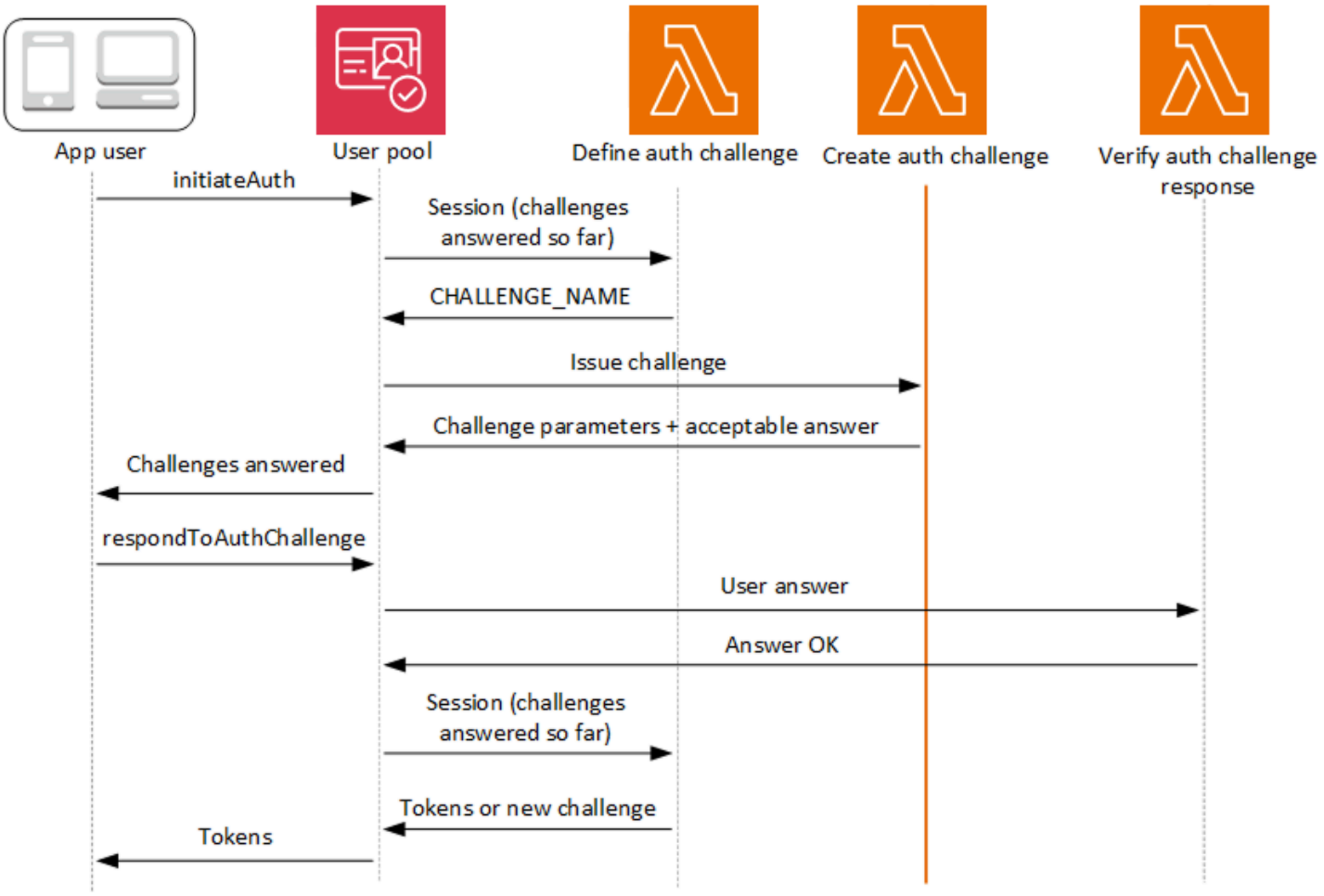
Node.js

```
const handler = async (event) => {
  if (
    event.request.session.length === 1 &&
    event.request.session[0].challengeName === "SRP_A"
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "PASSWORD_VERIFIER";
  } else if (
    event.request.session.length === 2 &&
    event.request.session[1].challengeName === "PASSWORD_VERIFIER" &&
    event.request.session[1].challengeResult === true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length === 3 &&
    event.request.session[2].challengeName === "CUSTOM_CHALLENGE" &&
```

```
    event.request.session[2].challengeResult === true
  ) {
    event.response.issueTokens = false;
    event.response.failAuthentication = false;
    event.response.challengeName = "CUSTOM_CHALLENGE";
  } else if (
    event.request.session.length === 4 &&
    event.request.session[3].challengeName === "CUSTOM_CHALLENGE" &&
    event.request.session[3].challengeResult === true
  ) {
    event.response.issueTokens = true;
    event.response.failAuthentication = false;
  } else {
    event.response.issueTokens = false;
    event.response.failAuthentication = true;
  }

  return event;
};

export { handler };
```

创建身份验证质询 Lambda 触发器

创建身份验证质询触发器是一个 Lambda 函数，其中包含由定义身份验证质询触发器声明的每个质询的详细信息。它处理由定义身份验证质询触发器声明的质询名称，并返回 `publicChallengeParameters`，您的应用程序必须将其呈现给用户。然后，此函数为您的用户池提供质询 `privateChallengeParameters` 的答案，您的用户池会将该质询传递给验证身份验证质询触发器。在您的定义身份验证质询触发器管理质询序列的地方，您的创建身份验证质询触发器管理质询内容。



创建身份验证质询

> 如果指定自定义质询作为定义身份验证质询 触发器的一部分，则 Amazon Cognito 会在定义身份验证质询之后调用此触发器。它将创建一个自定义身份验证流程。

系统调用此 Lambda 触发器来创建要向用户显示的质询。此 Lambda 触发器的请求包括 `challengeName` 和 `session`。`challengeName` 是一个字符串，是向用户显示的下一质询的名称。此属性的值在定义身份验证质询 Lambda 触发器中设置。

质询循环将一直重复到所有质询得到应答。

主题

- 创建身份验证质询 Lambda 触发器参数
- 创建身份验证质询示例

创建身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "challengeName": "string",
        "session": [
            ChallengeResult,
            . . .
        ],
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "publicChallengeParameters": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeMetadata": "string"
    }
}
```

创建身份验证质询请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

userNotFound

当为您的用户池客户端将 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，将填充此布尔值。

challengeName

新质询的名称。

会话

会话元素是一组 `ChallengeResult` 元素，其中，每个元素包含以下元素：

challengeName

质询类型。以下值之一：`"CUSTOM_CHALLENGE"`、`"PASSWORD_VERIFIER"`、`"SMS_MFA"`、`"DEVICE_SRP_AUTH"`、`"D` 或 `"ADMIN_NO_SRP_AUTH"`。

challengeResult

如果用户成功完成质询，则设置为 `true`，否则设置为 `false`。

challengeMetadata

您的自定义质询的名称。仅当 `challengeName` 为 `"CUSTOM_CHALLENGE"` 时使用。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为创建身份验证质询触发器指定的 Lambda 函数。您可以使用AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 操作中的 ClientMetadata 参数将此数据传递给您的 Lambda 函数。调用 create auth 质询函数的请求不包括在 API 操作中的 ClientMetadata AdminInitiateAuth参数中传递的数据。InitiateAuth

创建身份验证质询响应参数

publicChallengeParameters

客户端应用程序要在向用户显示的质询中使用的一个或多个键/值对。此参数应包含所有必要信息，以向用户准确显示质询。

privateChallengeParameters

此参数仅由验证身份验证质询响应 Lambda 触发器使用。此参数应包含所需的所有信息，以验证用户对质询的响应。也就是说，`publicChallengeParameters` 参数包含向用户显示的问题，`privateChallengeParameters` 包含问题的有效答案。

challengeMetadata

您的自定义质询的名称（如果是自定义质询）。

创建身份验证质询示例

此函数有两个自定义挑战，它们对应于我们定义身份验证挑战示例中的挑战序列。前两个挑战是 SRP 身份验证。对于第三个挑战，此函数在质询响应中向您的应用程序返回一个验证码网址。您的应用程序在给定的 URL 处呈现 CAPTCHA 并返回用户的输入。CAPTCHA 图像的 URL 作为 `captchaUrl` 添加到公有质询参数中，并且预期答案添加到私有质询参数中。

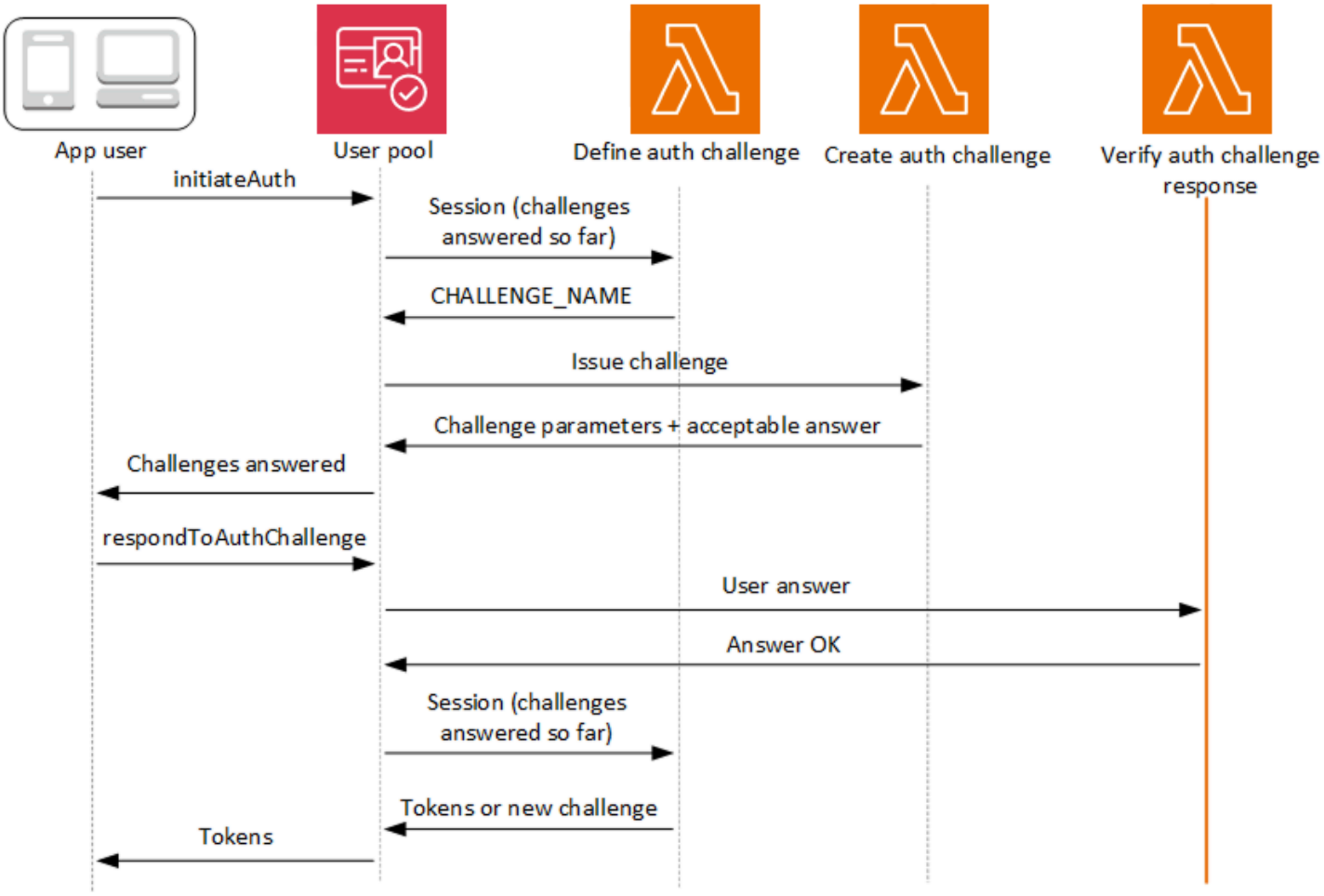
对于第四个挑战，此函数返回一个安全问题。您的应用程序会呈现问题并提示用户输入答案。在用户解决了两个自定义挑战后，define auth 质询触发器会确认您的用户池可以发行代币。

Node.js

```
const handler = async (event) => {
  if (event.request.challengeName !== "CUSTOM_CHALLENGE") {
    return event;
  }

  if (event.request.session.length === 2) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.captchaUrl = "url/123.jpg";
    event.response.privateChallengeParameters.answer = "5";
  }

  if (event.request.session.length === 3) {
    event.response.publicChallengeParameters = {};
    event.response.privateChallengeParameters = {};
    event.response.publicChallengeParameters.securityQuestion =
      "Who is your favorite team mascot?";
    event.response.privateChallengeParameters.answer = "Peccy";
  }

  return event;
};

export { handler };
```

验证身份验证质询响应 Lambda 触发器

验证身份验证质询触发器是一个 Lambda 函数，用于将用户提供的响应与已知回答进行比较。此功能告诉您的用户池，用户是否正确回答了质询。当验证身份验证质询触发器对 `answerCorrect` 的响应为 `true` 时，身份验证序列可以继续。



验证身份验证质询响应

> Amazon Cognito 调用此触发器，以验证用户对自定义身份验证质询的响应是否有效。它是用户池自定义身份验证流程的一部分。

此触发器的请求包括 `privateChallengeParameters` 和 `challengeAnswer` 参数。创建身份验证质询 Lambda 触发器返回 `privateChallengeParameters` 值，并包含用户的预期响应。`challengeAnswer` 参数包含用户对质询的响应。

响应包含 `answerCorrect` 属性。如果用户成功完成质询，Amazon Cognito 会将属性值设置为 `true`。如果用户未成功完成质询，Amazon Cognito 会将属性值设置为 `false`。

质询循环将一直重复，直至用户应答所有质询。

主题

- 验证身份验证质询 Lambda 触发器参数
- 验证身份验证质询响应示例

验证身份验证质询 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "privateChallengeParameters": {
            "string": "string",
            . . .
        },
        "challengeAnswer": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        },
        "userNotFound": boolean
    },
    "response": {
        "answerCorrect": boolean
    }
}
```

验证身份验证质询请求参数

userAttributes

此参数包含表示用户属性的一个或多个名称/值对。

userNotFound

当 Amazon Cognito 将您用户池客户端的 `PreventUserExistenceErrors` 设置为 `ENABLED` 时，Amazon Cognito 将填充此布尔值。

privateChallengeParameters

此参数来自创建身份验证质询触发器。为了确定用户是否通过了质询，Amazon Cognito 将参数与用户的 challengeAnswer 进行比较。

此参数包含所需的所有信息，以验证用户对质询的响应。该信息包括 Amazon Cognito 向用户提出的问题 (`publicChallengeParameters`)，以及问题的有效回答 (`privateChallengeParameters`)。只有验证身份验证质询响应 Lambda 触发器使用此参数。

challengeAnswer

此参数值是来自用户对质询响应的应答。

clientMetadata

此参数包含一个或多个键值对，您可以将其作为自定义输入内容提供给用于验证身份验证质询触发器的 Lambda 函数。要将此数据传递给您的 Lambda 函数，请使用AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 操作中的 ClientMetadata 参数。Amazon Cognito 在传递给验证身份验证质询函数的请求中不包含来自 ClientMetadata 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。

验证身份验证质询响应参数

answerCorrect

如果用户成功完成质询，Amazon Cognito 将此参数设置为 `true`。如果用户未成功完成质询，Amazon Cognito 将此参数设置为 `false`。

验证身份验证质询响应示例

此 verify auth 质询功能可检查用户对质询的响应是否与预期的响应相匹配。用户的答案由您的应用程序的输入定义，首选答案由创建身份验证质询触发器响应的响应`privateChallengeParameters.answer`中定义。正确答案和给定答案都是此函数输入事件的一部分。

在此示例中，如果用户的响应与预期的响应相匹配，则 Amazon Cognito 会将`answerCorrect`参数设置为。`true`

Node.js

```
const handler = async (event) => {
  if (
    event.request.privateChallengeParameters.answer ===
    event.request.challengeAnswer
  ) {
    event.response.answerCorrect = true;
  } else {
    event.response.answerCorrect = false;
  }

  return event;
};

export { handler };
```

令牌生成前 Lambda 触发器

由于 Amazon Cognito 会在令牌生成之前调用此触发器，您可以自定义用户池令牌中的声明。使用版本 1 或 `V1_0` 令牌生成前触发器事件的基本功能，可以自定义身份（ID）令牌。在具有 Essentials 或 Plus 功能计划的用户池中，您可以使用访问令牌自定义生成版本二或`V2_0`触发事件，以及通过 machine-to-machine (M2M) 客户端凭证授予的访问令牌自定义来生成版本三或`V3_0`触发事件。

Amazon Cognito 将向您的函数发送 `V1_0` 请求，其中包含将写入 ID 令牌的数据。`V2_0`或`V3_0`事件是指包含由 Amazon Cognito 写入身份令牌和访问令牌的数据的单个请求。要自定义这两个令牌，您必须更新函数以使用触发器版本二或三，并在同一个响应中发送两个令牌的数据。

Amazon Cognito 将版本二的事件响应应用于通过用户身份验证获得的访问令牌，其中人类用户已向您的用户池提供了证书。第三版事件响应适用于来自用户身份验证和计算机身份验证的访问令牌，其中自动化系统使用应用程序客户端密钥来授权访问令牌请求。除了生成的访问令牌的情况外，版本二和版本三的事件是相同的。

在 Amazon Cognito 向您的应用程序发布身份和访问令牌之前，此 Lambda 触发器可以添加、删除和修改这些令牌中的某些声明。要使用此功能，可以从 Amazon Cognito 用户池控制台关联 Lambda 函数或通过 Amazon Command Line Interface（Amazon CLI）更新用户池 `LambdaConfig`。

事件版本

您的用户池可以向您的 Lambda 函数提供不同版本的令牌生成前触发事件。`V1_0` 触发器提供用于修改 ID 令牌的参数。`V2_0`或`V3_0`触发器提供以下各项的参数。

1. `V1_0` 触发器的功能。
2. 能够自定义访问令牌。
3. 能够将复杂的数据类型传递给 ID 令牌和访问令牌声明值：
   - 字符串
   - 数字
   - 布尔值
   - 由字符串、数字、布尔值或它们的组合构成的数组
   - JSON

> **Note**
>
> 在 ID 令牌中，您可以将复杂对象填充到除了 `phone_number_verified`、`email_verified`、`updated_at` 和 `address` 之外的声明值。

默认情况下，用户池传送 `V1_0` 事件。要将您的用户池配置为发送`V2_0`事件，请在 Amazon Cognito 控制台中配置触发器时，选择基本功能的触发事件版本 +用户身份访问令牌自定义。要生成`V3_0`事件，请选择基本功能 + 用户和计算机身份的访问令牌自定义。您也可以在UpdateUserPool 或 CreateUserPool API 请求的`LambdaVersion`LambdaConfig参数中设置的值。活动版本一、二和三在 E ssential s 和 Plus 功能计划中可用。第三版活动的 M2M 操作的定价结构与月活跃用户 (MAU) 公式分开。有关更多信息，请参阅 Amazon Cognito 定价。

> **Note**
>
> 在 2024 年 11 月 22 日 18:00（格林威治标准时间）当天或之前使用 “高级安全功能” 选项运行且仍处于精简版功能层的用户池可以访问令牌生成前触发器的第一和第二版事件版本。此传统级别中没有高级安全功能的用户池可以访问事件版本一。第三版仅在 Essentials 和 Plus 中可用。

声明和范围参考

Amazon Cognito 限制了您可以在访问令牌和身份令牌中添加、修改或隐藏的声明和范围。下表描述了您的 Lambda 函数可以修改和不能修改的声明，以及影响声明存在或价值的触发事件参数。

Claim	默认令牌类型	可以添加吗？	可以修改吗？	可以抑制吗？	事件参数-添加或修改	事件参数-抑制	身份类型	活动版本
任何不在用户池令牌架构中的声明	无	支持	是	不适用	claimsToAddOrOverride	claimsToSuppress	用户、机器 [1]	全部 [2]
scope	访问	支持	是	是	scopesToAdd	scopesToSuppress	用户、机器 [1]	v2_0, v3_0
cognito:groups	身份证、访问权限	支持	是	是	groupsToOverride	claimsToSuppress	User	全部 [2]
cognito:preferred_role	ID	支持	是	是	preferredRole	claimsToSuppress [3]	User	全部
cognito:roles	ID	支持	是	是	iamRolesToOverride	claimsToSuppress [3]	User	全部
cognito:username	ID	否	否	否	不适用	不适用	User	不适用
任何其他带有cognito前缀的索赔	无	否	否	否	不适用	不适用	不适用	不适用

Claim	默认令牌类型	可以添加吗？	可以修改吗？	可以抑制吗？	事件参数-添加或修改	事件参数-抑制	身份类型	活动版本
username	访问	否	否	否	不适用	不适用	User	v2_0, v3_0
sub	身份证、访问权限	否	否	否	不适用	不适用	User	不适用
标准 OIDC 属性	ID	支持	是	是	claimsToAddOrOverride	claimsToSuppress	User	全部
custom: 属性	ID	支持	是	是	claimsToAddOrOverride	claimsToSuppress	User	全部
dev: 属性	ID	否	否	是	不适用	claimsToSuppress	User	全部
identities	ID	否	否	否	不适用	不适用	User	不适用
aud[4]	ID	否	否	否	不适用	不适用	用户、机器	不适用
client_id	访问	否	否	否	不适用	不适用	用户、机器	不适用
event_id	访问	否	否	否	不适用	不适用	用户、机器	不适用
device_key	访问	否	否	否	不适用	不适用	User	不适用
version	访问	否	否	否	不适用	不适用	用户、机器	不适用

Claim	默认令牌类型	可以添加吗？	可以修改吗？	可以抑制吗？	事件参数-添加或修改	事件参数-抑制	身份类型	活动版本
acr	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
amr	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
at_hash	ID	否	否	否	不适用	不适用	用户、机器	不适用
auth_time	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
azp	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
exp	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
iat	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用

Claim	默认令牌类型	可以添加吗？	可以修改吗？	可以抑制吗？	事件参数-添加或修改	事件参数-抑制	身份类型	活动版本
iss	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
jti	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
nbf	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
nonce	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
origin_ji	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用
token_use	身份证、访问权限	否	否	否	不适用	不适用	用户、机器	不适用

[1] 机器身份的访问令牌仅适用于触v3_0发器输入事件。活动版本三仅在 E ssential s 和 Plus 功能层中可用。精简版级别的用户池可以接收v1_0事件。具有高级安全功能的精简版用户池可以接收v1_0和v2_0事件。

[2] 将令牌生成前触发器配置`v1_0`为事件版本，仅适用`v2_0`于 ID 令牌、ID 和访问令牌、`v3_0`具有机器身份功能的 ID 和访问令牌。

[3] 要取消`cognito:preferred_role`和`cognito:roles`声明，请添加`cognito:groups`到`claimsToSuppress`。

[4] 您可以为访问令牌添加`aud`声明，但其值必须与当前会话的应用程序客户端 ID 相匹配。您可以从 `event.callerContext.clientId` 获取请求事件中的客户端 ID。

自定义身份令牌

使用令牌生成前 Lambda 触发器的所有事件版本，您可以自定义用户池中的身份 (ID) 令牌的内容。ID 令牌提供来自可信身份源的用户属性，用于登录 Web 或移动应用程序。有关 ID 令牌的更多信息，请参阅了解身份（ID）令牌。

将令牌生成前 Lambda 触发器与 ID 令牌结合使用，可实现以下目的。

- 在运行时更改您的用户从身份池中请求的 IAM 角色。
- 从外部来源添加用户属性。
- 添加或替换现有用户属性值。
- 隐藏用户属性，否则这些属性由于用户获得的授权范围以及您授予应用程序客户端的属性读取权限，会传递给您的应用程序。

自定义访问令牌

使用令牌生成前 Lambda 触发器的第二和第三个事件版本，您可以自定义用户池中访问令牌的内容。访问令牌授权用户从受访问保护的资源（例如 Amazon Cognito 令牌授权的 API 操作和第三方）中检索信息。 APIs对于使用客户凭证授予的 machine-to-machine（M2M）授权，Amazon Cognito 仅在您的用户池配置为版本三 () 事件时才会调用令牌生成前触发器。`V3_0`有关访问令牌的更多信息，请参阅了解访问令牌。

将令牌生成前 Lambda 触发器与访问令牌结合使用，可实现以下目的。

- 在`scope`声明中添加或隐藏作用域。例如，您可以将范围添加到由 Amazon Cognito 用户池 API 身份验证生成的访问令牌中，该身份验证仅分配范围 `aws.cognito.signin.user.admin`。
- 更改用户在用户池组中的成员资格。
- 添加尚不存在于 Amazon Cognito 访问令牌中的声明。

- 隐藏原本会传递到应用程序的声明。

要在用户池中支持访问自定义，必须将用户池配置为生成触发器请求的更新版本。请按照如下所示的过程更新用户池。

Amazon Web Services Management Console

在令牌生成前 Lambda 触发器中支持访问令牌自定义

1. 转到 Amazon Cognito 控制台，然后选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “扩展” 菜单并找到 Lambda 触发器。
4. 添加或编辑令牌生成前触发器。
5. 在分配 Lambda 函数下选择一个 Lambda 函数。
6. 选择触发事件版本的基本功能+用户身份的访问令牌自定义或基本功能+用户和计算机身份的访问令牌自定义。此设置会更新 Amazon Cognito 发送给您的函数的请求参数，使该函数包含用于自定义访问令牌的字段。

User pools API

在令牌生成前 Lambda 触发器中支持访问令牌自定义

生成CreateUserPool或 UpdateUserPoolAPI 请求。必须为所有您不想设置为默认值的参数指定一个值。有关更多信息，请参阅 更新用户池和应用程序客户端配置。

在请求的 `LambdaVersion` 参数中包含以下内容。`LambdaVersion`值为`V2_0`会使您的用户池为访问令牌添加参数并对其进行更改。`LambdaVersion`值为`V3_0`会生成与相同的事件`V2_0`，但会导致您的用户池也将更改应用于 M2M 访问令牌。要调用特定的函数版本，请使用以函数版本作为 `LambdaArn` 值的 Lambda 函数 ARN。

```
"PreTokenGenerationConfig": {
   "LambdaArn": "arn:aws:lambda:us-west-2:123456789012:function:MyFunction",
   "LambdaVersion": "V3_0"
},
```

machine-to-machine(M2M) 客户端凭证的客户端元数据

您可以在 M2M 请求中传递客户端元数据。客户机元数据是来自用户或应用程序环境的其他信息，这些信息可能有助于实现的结果令牌生成前 Lambda 触发器。在使用用户委托人的身份验证操作中，您可以将客户端元数据传递给 RespondToAuthChallengeAPI 请求正文中的令牌生成前触发器。AdminRespondToAuthChallenge由于应用程序通过直接向 M2M 请求来执行生成访问令牌的流程令牌端点，因此它们的模型不同。在客户端凭证令牌请求的 POST 正文中，将带有客户端元数据对象 URL-encoded (`x-www-form-urlencoded`) 的`aws_client_metadata`参数传递给字符串。有关示例请求，请参阅具有基本授权的客户凭证。以下是传递键值对的示例参数。`{"environment": "dev", "language": "en-US"}`

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US
%22%7D
```

更多资源

- 如何在 Amazon Cognito 用户池中自定义访问令牌

主题

- 令牌生成前 Lambda 触发器源
- 令牌生成前 Lambda 触发器参数
- 令牌生成前触发器事件版本 2 示例：添加和隐藏声明、范围及组
- 令牌生成前事件版本 2 示例：添加包含复杂对象的声明
- 令牌生成前事件版本 1 示例：添加新声明并隐藏现有声明
- 令牌生成前事件版本 1 示例：修改用户的组成员资格

令牌生成前 Lambda 触发器源

triggerSource 值	事件
`TokenGeneration_HostedAuth`	在身份验证期间从 Amazon Cognito 托管的登录登录页面调用。
`TokenGeneration_Authentication`	用户身份验证流完成之后调用。
`TokenGeneration_NewPassword Challenge`	管理员创建用户之后调用。当用户必须更改临时密码时调用此流。

triggerSource 值	事件
`TokenGeneration_ClientCredentials`	在 M2M 客户端凭证授予后调用。只有当您的活动版本为时，您的用户池才会发送此事件`V3_0`。
`TokenGeneration_AuthenticateDevice`	用户设备身份验证结束时调用。
`TokenGeneration_RefreshTokens`	用户尝试刷新身份和令牌时调用。

令牌生成前 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。在向用户池添加令牌生成前 Lambda 触发器时，您可以选择触发器版本。此版本决定 Amazon Cognito 是否将请求以及用于自定义访问令牌的附加参数传递给您的 Lambda 函数。

Version one

版本一令牌可以设置群组成员资格、IAM 角色和 ID 令牌中的新声明。群组成员资格优先权也适用于访问令牌中的`cognito:groups`索赔。

```
{
    "request": {
        "userAttributes": {"string": "string"},
        "groupConfiguration": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
        },
        "clientMetadata": {"string": "string"}
    },
    "response": {
        "claimsOverrideDetails": {
            "claimsToAddOrOverride": {"string": "string"},
```

```
            "claimsToSuppress": [
                "string",
                "string"
            ],
            "groupOverrideDetails": {
                "groupsToOverride": [
                    "string",
                    "string"
                ],
                "iamRolesToOverride": [
                    "string",
                    "string"
                ],
                "preferredRole": "string"
            }
        }
    }
}
```

Versions two and three

版本二和版本三的请求事件添加了自定义访问令牌的字段。用户池将版本三事件的更改应用于计算机身份的访问令牌。这些版本还增加了对响应对象中复杂`claimsToOverride`数据类型的支持。您的 Lambda 函数可以在 `claimsToOverride` 值中返回以下类型的数据：

- 字符串
- 数字
- 布尔值
- 由字符串、数字、布尔值或它们的组合构成的数组
- JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string"
        },
        "scopes": ["string", "string"],
        "groupConfiguration": {
            "groupsToOverride": ["string", "string"],
            "iamRolesToOverride": ["string", "string"],
```

```
            "preferredRole": "string"
        },
        "clientMetadata": {
            "string": "string"
        }
    },
    "response": {
        "claimsAndScopeOverrideDetails": {
            "idTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"]
            },
            "accessTokenGeneration": {
                "claimsToAddOrOverride": {
                    "string": [accepted datatype]
                },
                "claimsToSuppress": ["string", "string"],
                "scopesToAdd": ["string", "string"],
                "scopesToSuppress": ["string", "string"]
            },
            "groupOverrideDetails": {
                "groupsToOverride": ["string", "string"],
                "iamRolesToOverride": ["string", "string"],
                "preferredRole": "string"
            }
        }
    }
}
```

令牌生成前请求参数

名称	描述	最低触发器事件版本
userAttributes	用户池中用户配置文件的属性。	1
groupConfiguration	包含当前组配置的输入对象。对象包括 `groupsToOverride` 、`iamRolesToOverride` 和 `preferredRole` 。	1

名称	描述	最低触发器事件版本
groupsToOverride	您的用户所属的用户池组。	1
iamRolesTo覆盖	您可以将用户池组与 Amazon Identity and Access Management (IAM) 角色关联。此元素是您的用户所属组中的所有 IAM 角色的列表。	1
preferredRole	您可以为用户池组设置一个优先级。此元素包含 `groupsToOverride` 元素中具有最高优先级组中的 IAM 角色的名称。	1
clientMetadata	一个或多个键值对，您可以指定它们并将它们作为自定义输入提供给 Lambda 函数以用于令牌生成前的触发器。 要将此数据传递给您的 Lambda 函数，请使用AdminRespondToAuthChallenge和 RespondToAuthChallengeAPI 操作中的 ClientMetadata参数。Amazon Cognito 在传递给令牌生成前函数的请求中不包含来自`ClientMetadata` 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。	1
范围	访问令牌范围。访问令牌中的范围是您的用户请求且您授权应用程序客户端发布的用户池标准范围和自定义范围。	2

令牌生成前响应参数

名称	描述	最低触发器事件版本
claimsOverrideDetails	用于存放 `V1_0` 触发器事件中所有元素的容器。	1
claimsAndScopeOverrideDetails	`V2_0`或`V3_0`触发器事件中所有元素的容器。	2

名称	描述	最低触发器事件版本
idTokenGeneration	您要在用户的 ID 令牌中覆盖、添加或隐藏的声明。此 ID 令牌的父级自定义值仅出现在事件版本 2 及更高版本中，但子元素出现在版本 1 的事件中。	2
accessTokenGeneration	您要在用户的访问令牌中覆盖、添加或隐藏的声明和范围。访问令牌自定义值的父项仅出现在事件版本 2 及更高版本中。	2
claimsToAddOrOverride	您要添加或修改的一个或多个声明及其值的映射。对于与组相关的声明，请改用 `groupOverrideDetails` 。 在事件版本 2 及更高版本中，此元素同时出现在`accessTokenGeneration` 和之下`idTokenGeneration` 。	1*
claimsToSuppress	您希望 Amazon Cognito 隐藏的声明列表。如果您的函数同时隐藏并替换了声明值，则 Amazon Cognito 会隐藏声明。 在事件版本 2 及更高版本中，此元素同时出现在`accessTokenGeneration` 和之下`idTokenGeneration` 。	1

名称	描述	最低触发器事件版本
groupOverrideDetails	包含当前组配置的输出对象。对象包括 `groupsToOverride` 、`iamRolesToOverride` 和 `preferredRole` 。 您的函数将 `groupOverrideDetails` 对象替换为您提供的对象。如果您在响应中提供空的或空对象，则 Amazon Cognito 将隐藏组。要保持现有组配置不变，请将请求的 `groupConfiguration` 对象的值复制到响应中的 `groupOverrideDetails` 对象。然后将其传回服务。 Amazon Cognito ID 令牌和访问令牌都包含 `cognito:groups` 声明。在访问令牌和 ID 令牌中，您的 `groupOverrideDetails` 对象将替换 `cognito:groups` 声明。组覆盖是版本 1 事件可以对访问令牌进行的唯一更改。	1
scopesToAdd	您要添加到用户访问令牌中的`scope`声明的范围列表。不能添加包含一个或多个空格字符的范围值。	2
scopesToSuppress	您要从用户访问令牌中的`scope`声明中移除的范围列表。	2

* 版本一事件的响应对象可以返回字符串。版本二和版本三事件的响应对象可以返回复杂的对象。

令牌生成前触发器事件版本 2 示例：添加和隐藏声明、范围及组

此示例对用户的令牌进行了以下修改。

1. 在 ID 令牌中将其 `family_name` 设置为 `Doe`。
2. 防止 `email` 和 `phone_number` 声明出现在 ID 令牌中。
3. 将其 ID 令牌 `cognito:roles` 声明设置为 `"arn:aws:iam::123456789012:role\/sns_callerA","arn:aws:iam::123456789012:role\/sns_callerC","arn:aws:iam::123456789012:role\/sns_callerB"`。

4. 将其 ID 令牌 `cognito:preferred_role` 声明设置为 `arn:aws:iam::123456789012:role/sns_caller`。
5. 将作用域 `openid`、`email` 和 `solar-system-data/asteroids.add` 添加到访问令牌中。
6. 隐藏访问令牌的作用域 `phone_number` 和 `aws.cognito.signin.user.admin`。
   删除 `phone_number` 可阻止从 `userInfo` 中检索用户的电话号码。删除 `aws.cognito.signin.user.admin` 可阻止用户通过 Amazon Cognito 用户池 API 请求读取和修改自己的个人资料。

   > **Note**
   > 只有当访问令牌中的剩余作用域包括 `openid` 和至少一个其他标准作用域时，从作用域中删除 `phone_number` 才会阻止检索用户的电话号码。有关更多信息，请参阅 关于范围。

7. 将其 ID 和访问令牌 `cognito:groups` 声明设置为 `"new-group-A","new-group-B","new-group-C"`。

JavaScript

```
export const handler = function(event, context) {
  event.response = {
    "claimsAndScopeOverrideDetails": {
      "idTokenGeneration": {
        "claimsToAddOrOverride": {
          "family_name": "Doe"
    },
        "claimsToSuppress": [
          "email",
          "phone_number"
    ]
      },
      "accessTokenGeneration": {
        "scopesToAdd": [
          "openid",
          "email",
          "solar-system-data/asteroids.add"
        ],
        "scopesToSuppress": [
          "phone_number",
          "aws.cognito.signin.user.admin"
        ]
```

```
      },
      "groupOverrideDetails": {
        "groupsToOverride": [
    "new-group-A",
    "new-group-B",
    "new-group-C"
  ],
        "iamRolesToOverride": [
          "arn:aws:iam::123456789012:role/new_roleA",
          "arn:aws:iam::123456789012:role/new_roleB",
          "arn:aws:iam::123456789012:role/new_roleC"
        ],
        "preferredRole": "arn:aws:iam::123456789012:role/new_role",
      }
    }
  };
  // Return to Amazon Cognito
  context.done(null, event);
};
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_Authentication",
    "region": "us-east-1",
    "userPoolId": "us-east-1_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED",
            "email_verified": "true",
            "phone_number_verified": "true",
```

```
            "phone_number": "+12065551212",
            "family_name": "Zoe",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1",
 "arn:aws:iam::123456789012:role/sns_caller2", "arn:aws:iam::123456789012:role/
sns_caller3"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller"]
        },
        "scopes": [
            "aws.cognito.signin.user.admin", "openid", "email", "phone"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

令牌生成前事件版本 2 示例：添加包含复杂对象的声明

此示例对用户的令牌进行了以下修改。

1. 将数字、字符串、布尔值和 JSON 类型的声明添加到 ID 令牌。这是版本二触发器事件可对 ID 令牌进行的唯一更改。
2. 将数字、字符串、布尔值和 JSON 类型的声明添加到访问令牌。
3. 将三个范围添加到访问令牌。
4. 隐藏 ID 和访问令牌中的`email`声明。
5. 隐藏访问令牌中的 `aws.cognito.signin.user.admin` 范围。

JavaScript

```
export const handler = function(event, context) {

    var scopes = ["MyAPI.read", "MyAPI.write", "MyAPI.admin"]
    var claims = {}
    claims["aud"]= event.callerContext.clientId;
    claims["booleanTest"] = false;
    claims["longTest"] = 9223372036854775807;
```

```
   claims["exponentTest"] = 1.7976931348623157E308;
   claims["ArrayTest"] = ["test", 9223372036854775807, 1.7976931348623157E308,
true];
   claims["longStringTest"] = "\{\
       \"first_json_block\": \{\
           \"key_A\": \"value_A\",\
           \"key_B\": \"value_B\"\
       \},\
       \"second_json_block\": \{\
           \"key_C\": \{\
               \"subkey_D\": [\
                   \"value_D\",\
                   \"value_E\"\
               ],\
               \"subkey_F\": \"value_F\"\
           \},\
           \"key_G\": \"value_G\"\
       \}\
   \}";
   claims["jsonTest"] = {
    "first_json_block": {
     "key_A": "value_A",
     "key_B": "value_B"
    },
    "second_json_block": {
     "key_C": {
      "subkey_D": [
       "value_D",
       "value_E"
      ],
      "subkey_F": "value_F"
     },
     "key_G": "value_G"
    }
   };
   event.response = {
       "claimsAndScopeOverrideDetails": {
           "idTokenGeneration": {
               "claimsToAddOrOverride": claims,
               "claimsToSuppress": ["email"]
           },
           "accessTokenGeneration": {
               "claimsToAddOrOverride": claims,
               "claimsToSuppress": ["email"],
```

```
                "scopesToAdd": scopes,
                "scopesToSuppress": ["aws.cognito.signin.user.admin"]
            }
        }
    };
    console.info("EVENT response\n" + JSON.stringify(event, (_, v) => typeof v === 
 'bigint' ? v.toString() : v, 2))
    console.info("EVENT response size\n" + JSON.stringify(event, (_, v) => typeof v 
 === 'bigint' ? v.toString() : v).length)
    // Return to Amazon Cognito
    context.done(null, event);
};
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
    "version": "2",
    "triggerSource": "TokenGeneration_HostedAuth",
    "region": "us-west-2",
    "userPoolId": "us-west-2_EXAMPLE",
    "userName": "JaneDoe",
    "callerContext": {
        "awsSdkVersion": "aws-sdk-unknown-unknown",
        "clientId": "1example23456789"
    },
    "request": {
        "userAttributes": {
            "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
            "cognito:user_status": "CONFIRMED"
            "email_verified": "true",
            "phone_number_verified": "true",
            "phone_number": "+12065551212",
            "email": "Jane.Doe@example.com"
        },
        "groupConfiguration": {
            "groupsToOverride": ["group-1", "group-2", "group-3"],
            "iamRolesToOverride": ["arn:aws:iam::123456789012:role/sns_caller1"],
            "preferredRole": ["arn:aws:iam::123456789012:role/sns_caller1"]
```

```
        },
        "scopes": [
            "aws.cognito.signin.user.admin",
            "phone",
            "openid",
            "profile",
            "email"
        ]
    },
    "response": {
        "claimsAndScopeOverrideDetails": []
    }
}
```

令牌生成前事件版本 1 示例：添加新声明并隐藏现有声明

此示例将版本 1 触发器事件与令牌生成前 Lambda 函数结合使用，以添加新声明并隐藏现有声明。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      claimsToAddOrOverride: {
        my_first_attribute: "first_value",
        my_second_attribute: "second_value",
      },
      claimsToSuppress: ["email"],
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的测试事件：由于该代码示例不处理任何请求参数，因

此您可以使用带有空请求的测试事件。有关常见请求参数的更多信息，请参阅用户池 Lambda 触发器事件。

JSON

```
{
  "request": {},
  "response": {}
}
```

令牌生成前事件版本 1 示例：修改用户的组成员资格

此示例将版本 1 触发器事件与令牌生成前 Lambda 函数结合使用，以修改用户的组成员资格。

Node.js

```
const handler = async (event) => {
  event.response = {
    claimsOverrideDetails: {
      groupOverrideDetails: {
        groupsToOverride: ["group-A", "group-B", "group-C"],
        iamRolesToOverride: [
          "arn:aws:iam::XXXXXXXXXXXX:role/sns_callerA",
          "arn:aws:iam::XXXXXXXXX:role/sns_callerB",
          "arn:aws:iam::XXXXXXXXXX:role/sns_callerC",
        ],
        preferredRole: "arn:aws:iam::XXXXXXXXXXX:role/sns_caller",
      },
    },
  };

  return event;
};

export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "request": {},
  "response": {}
}
```

迁移用户 Lambda 触发器

如果用户在使用密码登录时或在使用忘记密码流程时不在用户池中，Amazon Cognito 会调用此触发器。Lambda 函数成功返回后，Amazon Cognito 将在用户池中创建用户。有关利用用户迁移 Lambda 触发器进行身份验证流程的详细信息，请参阅利用用户迁移 Lambda 触发器导入用户。

要在用户登录时或在忘记密码流程中，将用户从您的现有用户目录迁移到 Amazon Cognito 用户池，请使用此 Lambda 触发器。

主题

- 迁移用户 Lambda 触发器源
- 迁移用户 Lambda 触发器参数
- 示例：使用现有密码迁移用户

迁移用户 Lambda 触发器源

triggerSource 值	事件
`UserMigration_Authentication` [1]	登录时的用户迁移。
`UserMigration_ForgotPassword`	在忘记密码流程中迁移用户。

[1] 当用户使用无密码登录进行身份验证时，Amazon Cognito 不会调用此触发器。

迁移用户 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "userName": "string",
    "request": {
        "password": "string",
        "validationData": {
            "string": "string",
            . . .
        },
        "clientMetadata": {
            "string": "string",
       . . .
        }
    },
    "response": {
        "userAttributes": {
            "string": "string",
            . . .
        },
        "finalUserStatus": "string",
        "messageAction": "string",
        "desiredDeliveryMediums": [ "string", . . .],
        "forceAliasCreation": boolean,
        "enableSMSMFA": boolean
    }
}
```

迁移用户请求参数

userName

用户在登录时输入的用户名。

password

用户在登录时输入的密码。Amazon Cognito 不会在由忘记密码流程发起的请求中发送此值。

validationData

一个或多个键/值对，包含用户的登录请求中的验证数据。要将此数据传递给您的 Lambda 函数，您可以使用InitiateAuth和 AdminInitiateAuthAPI 操作中的 ClientMetadata 参数。

clientMetadata

一个或多个键/值对，您可以将其作为自定义输入内容提供给迁移用户触发器的 Lambda 函数。要将此数据传递给您的 Lambda 函数，您可以使用AdminRespondToAuthChallenge和ForgotPasswordAPI 操作中的 ClientMetadata 参数。

迁移用户响应参数

userAttributes

该字段为必填。

该字段必须包含一个或多个名称/值对，Amazon Cognito 将其存储在用户池的用户配置文件中并用作用户属性。您可以同时包括标准的和自定义的用户属性。自定义属性需要使用 `custom:` 前缀，以便与标准属性区分开来。有关更多信息，请参阅自定义属性。

> **Note**
> 要在忘记密码流程中重置密码，用户必须拥有经过验证的电子邮件地址或经过验证的电话号码。Amazon Cognito 将包含重置密码代码的消息发送到用户属性中的电子邮件地址或电话号码。

Attributes	要求
创建用户池时标记为必需的所有属性	如果迁移过程中缺少任何必需的属性，Amazon Cognito 将使用默认值。
`username`	如果您在为用户池配置了用户名登录之外，还配置了别名登录，并且用户输入了有效的别名值作为用户名，则此属性是必需的。该别名值可以是电子邮件地址、首选用户名或电话号码。 如果请求和用户池满足别名要求，则函数的响应必须将收到的 `username` 参数分配给别名属性，此外，响应必须将您自己的值分配给 `username` 属性。如果您的用户池不符合所需的条件，无法将收到的 `username` 发送到别名，则响应中的 `username` 参数必须与请求完全匹配，否则就会被忽略。

Attributes	要求
	Note 在用户池中，username 必须唯一。

finalUserStatus

您可以将此参数设置为 CONFIRMED 以自动确认用户，这样他们就可以使用之前的密码登录。当您将用户设置为 CONFIRMED 时，他们无需执行额外的操作即可登录。如果您未将此属性设置为 CONFIRMED，则它会设置为 RESET_REQUIRED。

若 finalUserStatus 设置为 RESET_REQUIRED，则意味着用户在迁移之后，必须在登录时立即更改密码，并且您的客户端应用程序必须在身份验证中处理 PasswordResetRequiredException。

> **Note**
>
> 使用 Lambda 触发器迁移期间，Amazon Cognito 不强制执行您为用户池配置的密码强度策略。如果密码不符合您配置的密码策略，Amazon Cognito 仍会接受密码，以便它继续迁移用户。要强制实施密码强度策略并拒绝不符合策略的密码，请验证代码中的密码强度。然后，如果密码不符合政策，则设置 finalUserStatus 为RESET_REQUIRED。

messageAction

您可以将此参数设置为 SUPPRESS，以拒绝发送 Amazon Cognito 通常会向新用户发送的欢迎消息。如果您的函数未返回此参数，Amazon Cognito 会发送欢迎消息。

desiredDeliveryMediums

您可以将此参数设置为 EMAIL 以通过电子邮件发送欢迎消息，或者设置为 SMS 以通过 SMS 发送欢迎消息。如果您的函数未返回此参数，Amazon Cognito 通过 SMS 发送欢迎消息。

forceAliasCreation

如果您将此参数设置为，TRUE并且 UserAttributes 参数中的电话号码或电子邮件地址已作为其他用户的别名存在，则 API 调用会将该别名从以前的用户迁移到新创建的用户。以前的用户无法再使用该别名登录。

如果您将此参数设置为 FALSE 而且别名存在，Amazon Cognito 不会迁移用户，并向客户端应用程序返回错误。

如果您不返回此参数，Amazon Cognito 会假定其值为“false”。

enableSMSMFA

将此参数设置为 true，要求迁移的用户完成 SMS 短信多重身份验证 (MFA) 才能登录。您的用户池必须启用 MFA。请求参数中的用户属性必须包含电话号码，否则该用户的迁移将失败。

示例：使用现有密码迁移用户

此示例 Lambda 函数使用现有密码迁移用户，并隐藏 Amazon Cognito 发送的欢迎消息。

Node.js

```
exports.handler = (event, context, callback) => {
  var user;

  if (event.triggerSource == "UserMigration_Authentication") {
    // authenticate the user with your existing user directory service
    user = authenticateUser(event.userName, event.request.password);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        email_verified: "true",
      };
      event.response.finalUserStatus = "CONFIRMED";
      event.response.messageAction = "SUPPRESS";
      context.succeed(event);
    } else {
      // Return error to Amazon Cognito
      callback("Bad password");
    }
  } else if (event.triggerSource == "UserMigration_ForgotPassword") {
    // Lookup the user in your existing user directory service
    user = lookupUser(event.userName);
    if (user) {
      event.response.userAttributes = {
        email: user.emailAddress,
        // required to enable password-reset code to be sent to user
        email_verified: "true",
      };
```

```
      event.response.messageAction = "SUPPRESS";
      context.succeed(event);
    } else {
      // Return error to Amazon Cognito
      callback("Bad password");
    }
  } else {
    // Return error to Amazon Cognito
    callback("Bad triggerSource " + event.triggerSource);
  }
};
```

自定义消息 Lambda 触发器

如果您对要发送给用户的电子邮件和短信设置了外部标准，或者您想在运行时将自己的逻辑应用于用户消息的格式化，请向您的用户池中添加自定义消息触发器。自定义消息 Lambda 会在您的用户池发送所有电子邮件和短信消息的内容之前先接收这些内容。然后，您的 Lambda 函数就有机会修改消息内容和主题。

Amazon Cognito 在发送电子邮件或电话验证消息或多重验证 (MFA) 代码前调用此触发器。您可以使用自定义消息触发器动态自定义消息。

该请求包括 `codeParameter`。这是一个字符串，用作 Amazon Cognito 传递给用户的代码的占位符。将 `codeParameter` 字符串插入消息正文中用于显示验证码的位置。Amazon Cognito 在收到此响应后，Amazon Cognito 将 `codeParameter` 字符串替换为实际验证码。

> **Note**
>
> 带有`CustomMessage_AdminCreateUser`触发源的自定义消息 Lambda 函数的输入事件包括用户名和验证码。由于管理员创建的用户必须同时收到其用户名和代码，因此函数的响应必须包含用户名和代码的占位符变量。消息的占位符是`request.usernameParameter`和`request.codeParameter`的值。这些值通常是`{username}`和`{####}`；作为最佳实践，请引用输入值，而不是对变量名称进行硬编码。

主题

- 自定义消息 Lambda 触发器源
- 自定义消息 Lambda 触发器参数

- 用于注册的自定义消息示例
- 管理员创建用户的自定义消息示例

自定义消息 Lambda 触发器源

triggerSource 值	事件
`CustomMessage_SignUp`	自定义消息 – 在注册后发送确认代码。
`CustomMessage_AdminCreateUser`	自定义消息 – 向新用户发送临时密码。
`CustomMessage_ResendCode`	自定义消息 – 向现有用户重新发送确认代码。
`CustomMessage_ForgotPassword`	自定义消息 – 针对“忘记密码”请求发送确认代码。
`CustomMessage_UpdateUserAttribute`	自定义消息 – 当用户的电子邮件或电话号码发生更改时，此触发器自动向用户发送验证码。不可用于其他属性。
`CustomMessage_VerifyUserAttribute`	自定义消息 – 当用户针对新的电子邮件或电话号码手动请求验证码时，此触发器向用户发送验证码。
`CustomMessage_Authentication`	自定义消息 – 在身份验证过程中发送 MFA 代码。

自定义消息 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "userAttributes": {
            "string": "string",
            . . .
```

```
        }
        "codeParameter": "####",
        "usernameParameter": "string",
        "clientMetadata": {
            "string": "string",
            . . .
        }
    },
    "response": {
        "smsMessage": "string",
        "emailMessage": "string",
        "emailSubject": "string"
    }
}
```

自定义消息请求参数

userAttributes

表示用户属性的一个或多个名称/值对。

codeParameter

一个字符串，用作自定义消息中验证码的占位符。

usernameParameter

用户名。Amazon Cognito 在管理员创建的用户发出的请求中包含此参数。

clientMetadata

一个或多个键值对，您可以将其作为自定义输入内容提供给为自定义消息触发器指定的 Lambda 函数。调用自定义消息函数的请求不包括在 InitiateAuthAPI 操作中的 ClientMetadata 参数中AdminInitiateAuth传递的数据。要将此数据传递给您的 Lambda 函数，您可以在以下 API 操作中使用 ClientMetadata 参数：

- AdminResetUserPassword
- AdminRespondToAuthChallenge
- AdminUpdateUserAttributes
- ForgotPassword
- GetUserAttributeVerificationCode
- ResendConfirmationCode

- SignUp
- UpdateUserAttributes

自定义消息响应参数

在响应中，指定要在发送给用户的消息中使用的自定义文本。有关 Amazon Cognito 适用于这些参数的字符串限制，请参阅。MessageTemplateType

smsMessage

要发送给用户的自定义 SMS 消息。必须包含您在请求中收到的 `codeParameter` 值。

emailMessage

发送给用户的自定义电子邮件。您可以在 `emailMessage` 参数中使用 HTML 格式。必须包含您在请求中收到的 `codeParameter` 值作为变量 `{####}`。只有在用户池的 `EmailSendingAccount` 属性为 `DEVELOPER` 时，Amazon Cognito 才可以使用`emailMessage` 参数。如果用户池的 `EmailSendingAccount` 属性不是 `DEVELOPER` 且返回了 `emailMessage` 参数，Amazon Cognito 会生成 400 错误代码 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`。当您选择使用 Amazon Simple Email Service (Amazon SES) 发送电子邮件时，用户池的 `EmailSendingAccount` 属性为 `DEVELOPER`。否则，该值为 `COGNITO_DEFAULT`。

emailSubject

自定义消息的主题行。只有当用户池的 EmailSendingAccount属性为时，您才能使用该`emailSubject`参数`DEVELOPER`。如果用户池的 `EmailSendingAccount` 属性不是 `DEVELOPER` 且 Amazon Cognito 返回了 `emailSubject` 参数，Amazon Cognito 会生成 400 错误代码 `com.amazonaws.cognito.identity.idp.model.InvalidLambdaResponseException`。当您选择使用 Amazon Simple Email Service (Amazon SES) 发送电子邮件时，用户池的 `EmailSendingAccount` 属性为 `DEVELOPER`。否则，该值为 `COGNITO_DEFAULT`。

用于注册的自定义消息示例

当服务要求应用程序向用户发送验证码时，此示例 Lambda 函数自定义电子邮件或 SMS 消息。

Amazon Cognito 可以在多个事件中调用 Lambda 触发器：注册后、重新发送验证码时、恢复忘记的密码时或验证用户属性时。响应包括电子邮件和 SMS 消息。该消息必须包含代码参数 `"####"`。此参数是用户收到的验证码的占位符。

电子邮件的最大长度为 20000 个 UTF-8 字符。此长度包括验证码。您可以在这些电子邮件中使用 HTML 标签。

SMS 消息的最大长度为 140 个 UTF-8 个字符。此长度包括验证码。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_SignUp") {
    const message = `Thank you for signing up. Your confirmation code is
 ${event.request.codeParameter}.`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service.";
  }
  return event;
};

export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
 "version": "1",
 "region": "us-west-2",
 "userPoolId": "us-west-2_EXAMPLE",
 "userName": "test-user",
 "callerContext": {
  "awsSdkVersion": "aws-sdk-unknown-unknown",
  "clientId": "1example23456789"
 },
 "triggerSource": "CustomMessage_SignUp",
 "request": {
  "userAttributes": {
   "sub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
   "cognito:user_status": "CONFIRMED",
   "email_verified": "true",
   "phone_number_verified": "true",
```

```
    "phone_number": "+12065551212",
    "email": "test-user@example.com"
   },
   "codeParameter": "{####}",
   "linkParameter": "{##Click Here##}",
   "usernameParameter": "None"
  },
  "response": {
   "smsMessage": "None",
   "emailMessage": "None",
   "emailSubject": "None"
  }
}
```

管理员创建用户的自定义消息示例

Amazon Cognito 向此示例自定义消息 Lambda 函数发送的请求的 `triggerSource` 值为 `CustomMessage_AdminCreateUser`，并且拥有用户名和临时密码。该函数根据请求中的临时密码填入 `${event.request.codeParameter}`，并根据请求中的用户名填入 `${event.request.usernameParameter}`。

您的自定义消息必须将 `codeParameter` 和 `usernameParameter` 的值插入响应对象中的 `smsMessage` 和 `emailMessage`。在此示例中，该函数将相同的消息写入响应字段 `event.response.smsMessage` 和 `event.response.emailMessage`。

电子邮件的最大长度为 20000 个 UTF-8 字符。此长度包括验证码。您可以在这些电子邮件中使用 HTML 标签。SMS 消息的最大长度为 140 个 UTF-8 个字符。此长度包括验证码。

响应包括电子邮件和 SMS 消息。

Node.js

```
const handler = async (event) => {
  if (event.triggerSource === "CustomMessage_AdminCreateUser") {
    const message = `Welcome to the service. Your user name is
 ${event.request.usernameParameter}. Your temporary password is
 ${event.request.codeParameter}`;
    event.response.smsMessage = message;
    event.response.emailMessage = message;
    event.response.emailSubject = "Welcome to the service";
  }
```

```
  return event;
};

export { handler };
```

Amazon Cognito 将事件信息传递给 Lambda 函数。随后，该函数将相同事件对象随同响应中的任何更改返回给 Amazon Cognito。在 Lambda 控制台中，您可以设置一个测试事件，该事件包含与您的 Lambda 触发器相关的数据。以下是此代码示例的一个测试事件：

JSON

```
{
  "version": 1,
  "triggerSource": "CustomMessage_AdminCreateUser",
  "region": "<region>",
  "userPoolId": "<userPoolId>",
  "userName": "<userName>",
  "callerContext": {
      "awsSdk": "<calling aws sdk with version>",
      "clientId": "<apps client id>",
      ...
  },
  "request": {
      "userAttributes": {
          "phone_number_verified": false,
          "email_verified": true,
           ...
      },
      "codeParameter": "####",
      "usernameParameter": "username"
  },
  "response": {
      "smsMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailMessage": "<custom message to be sent in the message with code parameter
 and username parameter>"
      "emailSubject": "<custom email subject>"
  }
}
```

自定义发件人 Lambda 触发器

Lambda 触发器 `CustomEmailSender` 和 `CustomSMSSender` 支持在用户池中使用第三方电子邮件和短信通知。您可以选择 SMS 和电子邮件提供商，通过 Lambda 函数代码向用户发送通知。当 Amazon Cognito 向用户发送邀请、MFA 代码、确认码、验证码或临时密码时，这些事件会激活您配置的 Lambda 函数。Amazon Cognito 将代码和临时密码（密钥）发送到您激活的 Lambda 函数。Amazon Cognito 使用 Amazon KMS 客户管理的密钥对这些机密进行加密，然后. Amazon Encryption SDK Amazon Encryption SDK 是一个客户端加密库，可帮助您加密和解密通用数据。

> **Note**
>
> 要将您的用户池配置为使用这些 Lambda 触发器，您可以使用 Amazon CLI 或软件开发工具包。无法从 Amazon Cognito 控制台进行这些配置。

CustomEmailSender

Amazon Cognito 调用此触发器向用户发送电子邮件通知。

自定义SMSSender

Amazon Cognito 调用此触发器向用户发送 SMS 通知。

所需的 资源

Amazon Cognito 不会在发送给自定义发件人触发器的事件中以明文形式发送用户的代码。Lambda 函数必须解密事件中的代码。以下概念是加密架构，您的函数必须使用该架构来获取可以传递给用户的代码。

Amazon KMS

Amazon KMS 是一项用于创建和控制 Amazon KMS 密钥的托管服务。这些密钥加密您的数据。有关更多信息，请参阅什么是 Amazon Key Management Service？。

KMS 密钥

KMS 密钥是加密密钥的逻辑表示形式。KMS 密钥包含元数据，如密钥 ID、创建日期、描述和密钥状态。KMS 密钥还包含用于加密和解密数据的密钥材料。有关更多信息，请参阅 Amazon KMS 密钥。

对称 KMS 密钥

对称 KMS 密钥是一个 256 位加密密钥，它不会退出 Amazon KMS 而不加密。要使用对称 KMS 密钥，必须调用 Amazon KMS。Amazon Cognito 使用对称密钥。加密和解密使用同一密钥。有关更多信息，请参阅对称 KMS 密钥。

自定义电子邮件发件人 Lambda 触发器

当您为用户群体分配自定义电子邮件发件人触发器时，如果用户事件要求 Amazon Cognito 发送电子邮件，则它会调用 Lambda 函数，而不是其原定设置行为。使用自定义发件人触发器，您的 Amazon Lambda 函数可以通过您选择的方法和提供商向用户发送电子邮件通知。您的函数的自定义代码必须处理和传递用户群体中的所有电子邮件。

此触发器适用于以下场景：您可能希望更好地控制用户池发送电子邮件消息的方式。您的 Lambda 函数可以自定义对 Amazon SES API 操作的调用，例如，当您想要管理多个经过验证的身份或跨 Amazon Web Services 区域时。您的函数还可能将消息重定向到另一个传递媒介或第三方服务。

> **Note**
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义发件人触发器。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 请求中使用 `LambdaConfig` 参数分配触发器。

要设置此触发器，请执行以下步骤：

1. 在 Amazon Key Management Service (Amazon KMS) 中创建对称加密密钥。Amazon Cognito 生成密钥（临时密码、验证码和确认码），然后使用此 KMS 密钥对这些密钥进行加密。然后，您可以在 Lambda 函数中使用解密 API 操作来解密这些密钥，并以明文形式将其发送给用户。Amazon Encryption SDK是在函数中进行 Amazon KMS 操作的有用工具。
2. 创建一个您要分配为自定义发件人触发器的 Lambda 函数。将您的 KMS 密钥的 `kms:Decrypt` 权限授予 Lambda 函数角色。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。
4. 编写 Lambda 函数代码，将您的消息定向到自定义传递方法或第三方提供商。要传递用户的验证码或确认码，请对请求中 `code` 参数的值进行 Base64 解码和解密。此操作将生成必须包含在消息中的明文代码或密码。

5. 更新用户池，使其使用自定义发件人 Lambda 触发器。使用自定义发件人触发器更新或创建用户群体的 IAM 主体必须具有为您的 KMS 密钥创建授予的权限。以下 `LambdaConfig` 代码段分配自定义短信和电子邮件发件人函数。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自定义电子邮件发件人 Lambda 触发器源

下表显示了 Lambda 代码中自定义电子邮件触发器源的触发事件。

TriggerSource value	事件
`CustomEmailSender_SignUp`	用户注册，Amazon Cognito 随即发送欢迎消息。
`CustomEmailSender_Authentication`	用户会登录，且 Amazon Cognito 会发送多重身份验证（MFA）代码。
`CustomEmailSender_ForgotPassword`	用户请求代码以重置其密码。
`CustomEmailSender_ResendCode`	用户请求替换账号确认代码。
`CustomEmailSender_UpdateUserAttribute`	用户更新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
`CustomEmailSender_VerifyUserAttribute`	用户创建新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
`CustomEmailSender_AdminCreateUser`	您在用户池中创建新用户，而 Amazon Cognito 向其发送临时密码。

TriggerSource value	事件
CustomEmailSender_AccountTakeOverNotification	Amazon Cognito 检测到接管用户账户的尝试并向用户发送通知。

自定义电子邮件发件人 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "type": "customEmailSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

自定义电子邮件发件人请求参数

type

请求版本。对于自定义电子邮件发件人事件，此字符串的值始终为 `customEmailSenderRequestV1`。

code

您的函数可以解密并发送给您的用户的加密代码。

clientMetadata

一个或多个键值对，您可以将它们作为自定义输入提供给自定义电子邮件发件人 Lambda 函数触发器。要将此数据传递给您的 Lambda 函数，您可以使用AdminRespondToAuthChallenge和

RespondToAuthChallengeAPI 操作中的 ClientMetadata 参数。Amazon Cognito 在传递给身份验证后函数的请求中不包含来自 ClientMetadata 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。

> **Note**
>
> 在具有以下触发源的事件中，Amazon Cognito 会向自定义电子邮件触发函数发送 `ClientMetadata`：
>
> - `CustomEmailSender_ForgotPassword`
> - `CustomEmailSender_SignUp`
> - `CustomEmailSender_Authentication`
>
> Amazon Cognito 不会在具有源 `CustomEmailSender_AccountTakeOverNotification` 的触发事件中发送 `ClientMetadata`。

userAttributes

表示用户属性的一个或多个键值对。

自定义电子邮件发件人响应参数

Amazon Cognito 不需要自定义电子邮件发件人响应中有任何其他返回信息。您的 Lambda 函数必须解释事件并解密代码，然后传送消息内容。典型的函数会组装电子邮件消息并将其定向到第三方 SMTP 中继。

激活自定义电子邮件发件人 Lambda 触发器

要设置使用自定义逻辑为您的用户池发送电子邮件消息的自定义电子邮件发件人触发器，请按如下方式激活触发器。以下步骤会将自定义电子邮件触发器和/或自定义 SMS 触发器分配给您的用户群体。在您添加自定义电子邮件发件人触发器后，Amazon Cognito 将始终向您的 Lambda 函数发送用户属性（包括电子邮件地址）和一次性代码，而本来会使用 Amazon Simple Email Service 发送电子邮件。

> **Important**
>
> Amazon Cognito HTML 会转义用户临时密码中的保留字符，例如 <（`<`）和 `>`（>）等。这些字符可能出现在 Amazon Cognito 发送到您的自定义电子邮件发件人函数的临时密码

中，但不会出现在临时验证码中。要发送临时密码，您的 Lambda 函数在解密密码之后必须取消对这些字符的转义，然后再将消息发送给您的用户。

1. 在 Amazon KMS中创建加密密钥。此密钥加密 Amazon Cognito 生成的临时密码和授权代码。然后，您可以在自定义发件人 Lambda 函数中解密这些密钥，将其以明文形式发送给用户。
2. 创建或更新您的用户池的 IAM 委托人会针对 Amazon Cognito 用于加密代码的 KMS 密钥创建一次性授权。向该委托人授予您的 KMS 密钥的`CreateGrant`权限。为了使此示例 KMS 密钥策略生效，更新用户池的管理员必须使用 IAM 角色的假设角色会话登录。`arn:aws:iam::111222333444:role/my-example-role`

   将以下基于资源的策略应用于您的 KMS 密钥。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111222333444:role/my-example-role"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 为自定义发件人触发器创建 Lambda 函数。Amazon Cognito 使用 Amazon Encryption SDK 加密密钥、临时密码和授权用户 API 请求的代码。
   - 为您的 Lambda 函数分配一个 IAM 角色，该角色至少具有对您的 KMS 密钥的 `kms:Decrypt` 权限。

4. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。

   以下 Amazon CLI 命令授予 Amazon Cognito 调用您的 Lambda 函数的权限：

```
aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 编写 Lambda 函数代码以发送消息。在 Amazon Cognito 将机密发送 Amazon Encryption SDK 给自定义发件人 Lambda 函数之前，Amazon Cognito 使用它来加密机密。在您的函数中，解密密钥并处理任何相关的元数据。然后将代码、您自己的自定义消息和目标电话号码发送到传送消息的自定义 API。

6. 将 Amazon Encryption SDK 添加到您的 Lambda 函数中。有关更多信息，请参阅 Amazon Encryption SDK 编程语言。要更新 Lambda 包，请完成以下步骤：

   a. 在 Amazon Web Services Management Console中将您的 Lambda 函数作为.zip 文件导出。

   b. 打开您的函数并添加 Amazon Encryption SDK. 有关更多信息和下载链接，请参阅 Amazon Encryption SDK Developer Guide（《Crypto SDK 开发人员指南》）中的 Amazon Encryption SDK 编程语言。

   c. 压缩您的函数及 SDK 依赖项，然后将函数上传到 Lambda。有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的将 Lambda 函数部署为 .zip 文件归档。

7. 更新用户群体，添加自定义发件人 Lambda 触发器。在 `UpdateUserPool` API 请求中包含 `CustomSMSSender` 或 `CustomEmailSender` 参数。`UpdateUserPool` API 操作需要更新用户群体的所有参数和 您要更改的参数。如果您没有提供所有相关参数，Amazon Cognito 会将任何缺失参数的值设置为其原定设置。如以下示例所示，包括您想要添加到或保留在用户群体中的所有 Lambda 函数的条目。有关更多信息，请参阅 更新用户池和应用程序客户端配置。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations. This snippet also includes a pre sign-up trigger for
syntax reference. The pre sign-up trigger
   #doesn't have a role in custom sender triggers.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
```

```
                        CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                        KMSKeyID=key-id"
```

要删除带有的自定义发件人 Lambda 触发器 `update-user-pool` Amazon CLI，请省略`CustomSMSSender`或`CustomEmailSender`参数`--lambda-config`，并包含您要在用户池中使用的所有其他触发器。

要使用 `UpdateUserPool` API 请求删除自定义发件人 Lambda 触发器，请在包含其余用户群体配置的请求正文中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数。

代码示例

以下 Node.js 示例在您的自定义电子邮件发件人 Lambda 函数中处理电子邮件事件。此示例假设您的函数定义了两个环境变量。

KEY_ALIAS

要用于加密和解密用户代码的 KMS 密钥的别名。

KEY_ARN

要用于加密和解密用户代码的 KMS 密钥的 Amazon 资源名称 (ARN)。

部署此函数

1. 在您的开发者工作区中安装最新版本的 NodeJS。
2. 在您的工作空间中创建一个新的 NodeJS 项目。
3. 使用初始化您的项目`npm init -y`。
4. 为 Lambda 函数创建脚本:。`touch index.mjs`
5. 将以下示例的内容粘贴到`index.mjs`。
6. 下载项目依赖关系， Amazon Encryption SDK:`npm install @aws-crypto/client-node`.
7. 将项目目录压缩成一个文件:`zip -r my_deployment_package.zip .`.
8. 将 ZIP 文件部署到您的函数。

此示例函数对代码进行解密，对于注册事件，它模拟向用户的电子邮件地址发送电子邮件。

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-
node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
    CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);

const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending email.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an email message to 'emailaddress'
 with body 'message'.
const sendEmail = async (emailAddress, message) => {
    // Log the destination with the email address masked.
    console.log(`Simulating email send to ${emailAddress.replace(/[^@.]/g, '*')}`);
    // Log the message with the code masked.
    console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '********')}`);
    // Simulate API delay
    await new Promise(resolve => setTimeout(resolve, 100));
    console.log('Email sent successfully');
    return true;
};

export const handler = async (event) => {
    try {
        // Decrypt the secret code using encryption SDK
        let plainTextCode;
        if (event.request.code) {
            const { plaintext, messageHeader } = await decrypt(keyring,
 Buffer.from(event.request.code, 'base64'));
            plainTextCode = Buffer.from(plaintext).toString('utf-8');
        }

        // Handle different trigger sources
        if (event.triggerSource == 'CustomEmailSender_SignUp') {
            const emailAddress = event.request.userAttributes.email;
            const message = `Welcome! Your verification code is: ${plainTextCode}`;
            await sendEmail(emailAddress, message);
        }
```

```
        else if (event.triggerSource == 'CustomEmailSender_ResendCode') {
            // Handle resend code
        }
        else if (event.triggerSource == 'CustomEmailSender_ForgotPassword') {
            // Handle forgot password
        }
        else if (event.triggerSource == 'CustomEmailSender_UpdateUserAttribute') {
            // Handle update attribute
        }
        else if (event.triggerSource == 'CustomEmailSender_VerifyUserAttribute') {
            // Handle verify attribute
        }
        else if (event.triggerSource == 'CustomEmailSender_AdminCreateUser') {
            // Handle admin create user
        }
        else if (event.triggerSource == 'CustomEmailSender_Authentication') {
            // Handle authentication
        }
        else if (event.triggerSource ==
 'CustomEmailSender_AccountTakeOverNotification') {
            // Handle account takeover notification
        }

        return;
    } catch (error) {
        console.error('Error in custom email sender:', error);
        throw error;
    }
};
```

自定义 SMS 发件人 Lambda 触发器

当您为用户群体分配自定义短信发件人触发器时，如果用户事件要求 Amazon Cognito 发送短信，则它会调用 Lambda 函数，而不是其原定设置行为。使用自定义发件人触发器，您的 Amazon Lambda 函数可以通过您选择的方法和提供商向用户发送 SMS 通知。您的函数的自定义代码必须处理和传递用户群体中的所有短信。

此触发器适用于以下场景：您可能希望更好地控制用户池发送短信消息的方式。您的 Lambda 函数可以自定义对 Amazon SNS API 操作的调用，例如，当您想要管理多个 IDs 起源或交叉时。 Amazon Web Services 区域您的函数还可能将消息重定向到另一个传递媒介或第三方服务。

> **Note**
>
> 目前，您无法在 Amazon Cognito 控制台中分配自定义发件人触发器。您可以在 `CreateUserPool` 或 `UpdateUserPool` API 请求中使用 `LambdaConfig` 参数分配触发器。

要设置此触发器，请执行以下步骤：

1. 在 Amazon Key Management Service (Amazon KMS) 中创建对称加密密钥。Amazon Cognito 生成密钥（临时密码、验证码和确认码），然后使用此 KMS 密钥对这些密钥进行加密。然后，您可以在 Lambda 函数中使用解密 API 操作来解密这些密钥，并以明文形式将其发送给用户。Amazon Encryption SDK是对函数进行 Amazon KMS 操作的有用工具。
2. 创建一个您要分配为自定义发件人触发器的 Lambda 函数。将您的 KMS 密钥的 `kms:Decrypt` 权限授予 Lambda 函数角色。
3. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。
4. 编写 Lambda 函数代码，将您的消息定向到自定义传递方法或第三方提供商。要传递用户的验证码或确认码，请对请求中 `code` 参数的值进行 Base64 解码和解密。此操作将生成必须包含在消息中的明文代码或密码。
5. 更新用户池，使其使用自定义发件人 Lambda 触发器。使用自定义发件人触发器更新或创建用户群体的 IAM 主体必须具有为您的 KMS 密钥创建授予的权限。以下 `LambdaConfig` 代码段分配自定义短信和电子邮件发件人函数。

```
"LambdaConfig": {
   "KMSKeyID": "arn:aws:kms:us-
east-1:123456789012:key/a6c4f8e2-0c45-47db-925f-87854bc9e357",
   "CustomEmailSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   },
   "CustomSMSSender": {
      "LambdaArn": "arn:aws:lambda:us-east-1:123456789012:function:MyFunction",
      "LambdaVersion": "V1_0"
   }
```

自定义 SMS 发件人 Lambda 触发器源

下表显示了 Lambda 代码中自定义 SMS 触发器源的触发事件。

TriggerSource value	事件
CustomSMSSender_SignUp	用户注册，Amazon Cognito 随即发送欢迎消息。
CustomSMSSender_ForgotPassword	用户请求代码以重置其密码。
CustomSMSSender_ResendCode	用户请求新代码来确认其注册。
CustomSMSSender_VerifyUserAttribute	用户创建新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
CustomSMSSender_UpdateUserAttribute	用户更新电子邮件地址或电话号码属性，而 Amazon Cognito 将发送代码用于验证该属性。
CustomSMSSender_Authentication	配置了 SMS 多重验证 (MFA) 的用户将登录。
CustomSMSSender_AdminCreateUser	您在用户池中创建新用户，而 Amazon Cognito 向其发送临时密码。

自定义 SMS 发件人 Lambda 触发器参数

Amazon Cognito 传递给此 Lambda 函数的请求是以下参数和 Amazon Cognito 添加到所有请求中的常用参数的组合。

JSON

```
{
    "request": {
        "type": "customSMSSenderRequestV1",
        "code": "string",
        "clientMetadata": {
            "string": "string",
             . . .
            },
        "userAttributes": {
            "string": "string",
            . . .
         }
}
```

自定义 SMS 发件人请求参数

type

请求版本。对于自定义 SMS 发件人事件，此字符串的值始终为 `customSMSSenderRequestV1`。

code

您的函数可以解密并发送给您的用户的加密代码。

clientMetadata

一个或多个键值对，您可以将它们作为自定义输入提供给自定义 SMS 发件人 Lambda 函数触发器。要将此数据传递给您的 Lambda 函数，您可以使用AdminRespondToAuthChallenge和RespondToAuthChallengeAPI 操作中的 ClientMetadata 参数。Amazon Cognito 在传递给身份验证后函数的请求中不包含来自 ClientMetadata 参数AdminInitiateAuth和 InitiateAuthAPI 操作的数据。

userAttributes

表示用户属性的一个或多个键值对。

自定义 SMS 发件人响应参数

Amazon Cognito 不需要响应中任何额外的返回信息。您的函数可以使用 API 操作来查询和修改资源，或者将事件元数据记录到外部系统。

激活自定义 SMS 发件人 Lambda 触发器

您可以设置自定义 SMS 发件人触发器，使用自定义逻辑为您的用户群体发送 SMS 消息。以下步骤会将自定义 SMS 触发器和/或自定义电子邮件触发器分配给您的用户群体。添加自定义 SMS 发件人触发器后，Amazon Cognito 始终向您的 Lambda 函数发送用户属性（包括电话号码）和一次性代码，而不是采用默认使用 Amazon Simple Notification Service 发送 SMS 消息的行为。

> **Important**
>
> Amazon Cognito HTML 会转义用户临时密码中的保留字符，例如 <（`<`）和 `>`（>）等。这些字符可能出现在 Amazon Cognito 发送到您的自定义电子邮件发件人函数的临时密码中，但不会出现在临时验证码中。要发送临时密码，您的 Lambda 函数在解密密码之后必须取消对这些字符的转义，然后再将消息发送给您的用户。

1. 在 Amazon KMS中创建加密密钥。此密钥加密 Amazon Cognito 生成的临时密码和授权代码。然后，您可以在自定义发件人 Lambda 函数中解密这些密钥，将其以明文形式发送给用户。
2. 创建或更新您的用户池的 IAM 委托人会针对 Amazon Cognito 用于加密代码的 KMS 密钥创建一次性授权。向该委托人授予您的 KMS 密钥的`CreateGrant`权限。为了使此示例 KMS 密钥策略生效，更新用户池的管理员必须使用 IAM 角色的假设角色会话登录。`arn:aws:iam::111222333444:role/my-example-role`

   将以下基于资源的策略应用于您的 KMS 密钥。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Principal": {
            "AWS": "arn:aws:iam::111222333444:role/my-example-role"
        },
        "Action": "kms:CreateGrant",
        "Resource": "arn:aws:kms:us-
west-2:111222333444:key/1example-2222-3333-4444-999example",
        "Condition": {
            "StringEquals": {
                "aws:SourceAccount": "111222333444"
            },
            "ArnLike": {
                "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111222333444:userpool/us-east-1_EXAMPLE"
            }
        }
    }]
}
```

3. 为自定义发件人触发器创建 Lambda 函数。Amazon Cognito 使用 Amazon Encryption SDK 加密密钥、临时密码和授权用户 API 请求的代码。
   - 为您的 Lambda 函数分配一个 IAM 角色，该角色至少具有对您的 KMS 密钥的 `kms:Decrypt` 权限。
4. 授予 Amazon Cognito 服务主体 `cognito-idp.amazonaws.com` 访问权限，以调用 Lambda 函数。

   以下 Amazon CLI 命令授予 Amazon Cognito 调用您的 Lambda 函数的权限：

```
aws lambda add-permission --function-name lambda_arn --statement-id
 "CognitoLambdaInvokeAccess" --action lambda:InvokeFunction --principal cognito-
idp.amazonaws.com
```

5. 编写 Lambda 函数代码以发送消息。在 Amazon Cognito 将机密发送 Amazon Encryption SDK 给自定义发件人 Lambda 函数之前，Amazon Cognito 使用它来加密机密。在您的函数中，解密密钥并处理任何相关的元数据。然后将代码、您自己的自定义消息和目标电话号码发送到传送消息的自定义 API。
6. 将 Amazon Encryption SDK 添加到您的 Lambda 函数中。有关更多信息，请参阅 Amazon Encryption SDK 编程语言。要更新 Lambda 包，请完成以下步骤：
   a. 在 Amazon Web Services Management Console中将您的 Lambda 函数作为.zip 文件导出。
   b. 打开您的函数并添加 Amazon Encryption SDK. 有关更多信息和下载链接，请参阅 Amazon Encryption SDK Developer Guide（《Crypto SDK 开发人员指南》）中的 Amazon Encryption SDK 编程语言。
   c. 压缩您的函数及 SDK 依赖项，然后将函数上传到 Lambda。有关更多信息，请参阅《Amazon Lambda 开发人员指南》中的将 Lambda 函数部署为 .zip 文件归档。
7. 更新用户群体，添加自定义发件人 Lambda 触发器。在 `UpdateUserPool` API 请求中包含 `CustomSMSSender` 或 `CustomEmailSender` 参数。`UpdateUserPool` API 操作需要更新用户群体的所有参数和 您要更改的参数。如果您没有提供所有相关参数，Amazon Cognito 会将任何缺失参数的值设置为其原定设置。如以下示例所示，包括您想要添加到或保留在用户群体中的所有 Lambda 函数的条目。有关更多信息，请参阅 更新用户池和应用程序客户端配置。

```
   #Send this parameter in an 'aws cognito-idp update-user-pool' CLI command,
including any existing
   #user pool configurations. This snippet also includes a pre sign-up trigger for
syntax reference. The pre sign-up trigger
   #doesn't have a role in custom sender triggers.

     --lambda-config "PreSignUp=lambda-arn, \
                      CustomSMSSender={LambdaVersion=V1_0,LambdaArn=lambda-arn}, \
                      CustomEmailSender={LambdaVersion=V1_0,LambdaArn=lambda-arn},
\
                      KMSKeyID=key-id"
```

要删除带有的自定义发件人 Lambda 触发器 `update-user-pool` Amazon CLI，请省略`CustomSMSSender`或`CustomEmailSender`参数`--lambda-config`，并包含您要在用户池中使用的所有其他触发器。

要使用 `UpdateUserPool` API 请求删除自定义发件人 Lambda 触发器，请在包含其余用户群体配置的请求正文中省略 `CustomSMSSender` 或 `CustomEmailSender` 参数。

代码示例

以下 Node.js 示例在您的自定义 SMS 发件人 Lambda 函数中处理 SMS 消息事件。此示例假设您的函数定义了两个环境变量。

KEY_ALIAS

要用于加密和解密用户代码的 KMS 密钥的别名。

KEY_ARN

要用于加密和解密用户代码的 KMS 密钥的 Amazon 资源名称 (ARN)。

部署此函数

1. 在您的开发者工作区中安装最新版本的 NodeJS。
2. 在您的工作空间中创建一个新的 NodeJS 项目。
3. 使用初始化您的项目`npm init -y`。
4. 为 Lambda 函数创建脚本:。`touch index.mjs`
5. 将以下示例的内容粘贴到`index.mjs`。
6. 下载项目依赖关系， Amazon Encryption SDK:`npm install @aws-crypto/client-node`.
7. 将项目目录压缩成一个文件:`zip -r my_deployment_package.zip .`.
8. 将 ZIP 文件部署到您的函数。

```
import { KmsKeyringNode, buildClient, CommitmentPolicy } from '@aws-crypto/client-
node';

// Configure the encryption SDK client with the KMS key from the environment variables
const { encrypt, decrypt } = buildClient(
    CommitmentPolicy.REQUIRE_ENCRYPT_ALLOW_DECRYPT
);
```

```
const generatorKeyId = process.env.KEY_ID;
const keyIds = [process.env.KEY_ARN];
const keyring = new KmsKeyringNode({ generatorKeyId, keyIds });

// Example function to simulate sending SMS.
// This example logs message details to CloudWatch Logs from your Lambda function.
// Update this function with custom logic that sends an SMS message to 'phoneNumber'
 with body 'message'.
const sendSMS = async (phoneNumber, message) => {
    // Log the destination with the phone number masked.
    console.log(`Simulating SMS send to ${phoneNumber.replace(/[^+]/g, '*')}`);
    // Log the message with the code masked.
    console.log(`Message content: ${message.replace(/\b\d{6,8}\b/g, '********')}`);
    // Simulate API delay
    await new Promise(resolve => setTimeout(resolve, 100));
    console.log('SMS sent successfully');
    return true;
};

export const handler = async (event) => {
    try {
        // Decrypt the secret code using encryption SDK
        let plainTextCode;
        if (event.request.code) {
            const { plaintext, messageHeader } = await decrypt(keyring,
 Buffer.from(event.request.code, 'base64'));
            plainTextCode = Buffer.from(plaintext).toString('utf-8');
        }

        // Handle different trigger sources
        if (event.triggerSource == 'CustomSMSSender_SignUp') {
            const phoneNumber = event.request.userAttributes.phone_number;
            const message = `Welcome! Your verification code is: ${plainTextCode}`;
            await sendSMS(phoneNumber, message);
        }
        else if (event.triggerSource == 'CustomSMSSender_ResendCode') {
            // Handle resend code
        }
        else if (event.triggerSource == 'CustomSMSSender_ForgotPassword') {
            // Handle forgot password
        }
        else if (event.triggerSource == 'CustomSMSSender_UpdateUserAttribute') {
            // Handle update attribute
        }
```

```
        else if (event.triggerSource == 'CustomSMSSender_VerifyUserAttribute') {
            // Handle verify attribute
        }
        else if (event.triggerSource == 'CustomSMSSender_AdminCreateUser') {
            // Handle admin create user
        }
        return;
    } catch (error) {
        console.error('Error in custom SMS sender:', error);
        throw error;
    }
};
```

主题

- 使用自定义 SMS 发件人函数评估 SMS 消息功能

使用自定义 SMS 发件人函数评估 SMS 消息功能

自定义 SMS 发件人 Lambda 函数接受您的用户池将发送的 SMS 消息，并且该函数根据您的自定义逻辑传送内容。Amazon Cognito 将 自定义 SMS 发件人 Lambda 触发器参数 发送到您的函数。您的函数可以用这些信息做您想做的事。例如，您可以将代码发送到 Amazon Simple Notification Service (Amazon SNS) 主题。Amazon SNS 主题订阅者可以是 SMS 消息、HTTPS 终端节点或电子邮件地址。

要使用自定义短信发送器 Lambda 函数为 Amazon Cognito 短信创建测试环境，请参阅上的 aws-sample amazon-cognito-user-pools development-and-testing-with 库sms-redirected-to-email中的--。GitHub存储库包含可以创建新用户池或使用已有的用户池的 Amazon CloudFormation 模板。这些模板创建 Lambda 函数和 Amazon SNS 主题。模板分配为自定义 SMS 发件人触发器的 Lambda 函数将您的 SMS 消息重定向到 Amazon SNS 主题的订阅者。

当您将此解决方案部署到用户池时，对于 Amazon Cognito 通常通过 SMS 消息发送的所有消息，Lambda 函数将改为发送到中央电子邮件地址。使用此解决方案自定义和预览 SMS 消息，并测试促使 Amazon Cognito 发送 SMS 消息的用户池事件。完成测试后，回滚 CloudFormation 堆栈，或者从用户池中移除自定义短信发送器功能分配。

> **Important**
>
> 不要使用 amazon-cognito-user-pool-development-and-testing-with- 中的模板sms-redirected-to-email来构建生产环境。解决方案中的自定义 SMS 发件人 Lambda 函数模拟 SMS 消息，但

Lambda 函数将它们全部发送到一个中央电子邮件地址。在生产 Amazon Cognito 用户池中发送 SMS 消息之前，您必须完成 Amazon Cognito 用户池的短信设置中显示的要求。

管理用户池中的用户

创建用户群体后，您可以创建、确认和管理用户账户。借助 Amazon Cognito 用户池组，您可以通过将 IAM 角色映射到组来管理您的用户及其对资源的访问。

管理 Amazon Cognito 用户池中的用户涉及各种配置选项和管理任务。用户池可以扩展到数百万用户。这种规模的用户目录需要同样可扩展和可重复的管理工具。您可能需要创建许多用户配置文件、管理不活跃的用户、生成治理和合规报告，或者设置自助工具，让用户自行处理大部分工作。创建用户池后，您可以控制用户如何注册和确认账户，包括要求验证电子邮件或电话号码。管理员还可以直接创建用户账户并自定义欢迎消息和密码要求。

用户池有用户组，您可以根据用户的组成员资格来管理对资源的访问权限。您可以为这些组分配 IAM 角色，从而使用身份池来管理对 Amazon Web Services 服务 的访问权限。ID 令牌和访问令牌中均存在用户的组成员资格。有了这些信息，您就可以在应用程序运行时或使用诸如 Amazon Verified Permissions 之类的策略引擎作出访问控制决策。

用户池通常有许多用户。您会发现自己经常需要搜索和更新用户账户。Amazon Cognito 控制台和 API 支持根据用户名、电子邮件和电话号码等标准属性查询用户。管理员还可以重置密码、禁用账户和查看用户事件历史记录。

为了迁移现有用户数据，Amazon Cognito 可以选择从 CSV 文件导入用户，也可以在用户首次登录时使用 Lambda 触发器自动迁移用户。这些选项支持用户从其他用户目录转换到您的用户池。

您可以使用用户池中的用户管理特征来精细控制用户生命周期和身份验证体验。自助注册、管理员创建的账户、组和迁移工具相结合，使 Amazon Cognito 用户池成为灵活的用户目录。

主题

- 配置用户创建策略
- 注册并确认用户账户
- 以管理员身份创建用户账户
- 向用户池添加组
- 管理和搜索用户账户
- 密码、账户恢复和密码策略

- 将用户导入一个用户池
- 使用用户属性

配置用户创建策略

您的用户池可以允许用户注册，您也可以以管理员身份创建用户。您还可以控制注册后的验证和确认过程由用户掌控的程度。例如，您可能需要审查注册并根据外部验证流程接受注册。此配置，或管理员创建用户策略，还可设置用户无法再确认其用户账户之前的时间长度。

作为软件的客户身份和访问管理（CIAM）平台，Amazon Cognito 可以满足公众客户的需求。接受注册并拥有应用程序客户端的用户池，无论是否使用托管登录，都会为互联网上任何知道您可公开发现的应用程序客户端 ID 并请求注册的人创建用户个人资料。注册的用户配置文件可接收访问和身份令牌，还可以访问您为应用程序授权的资源。在用户池中激活注册之前，请查看您的选项并确保配置符合您的安全标准。请按照以下步骤所述，谨慎设置启用自行注册和 `AllowAdminCreateUserOnly`。

Amazon Web Services Management Console

用户池的 “注册” 菜单包含一些用于在用户池中注册和管理用户创建的设置。

配置注册体验

1. 在 Cognito 辅助验证和确认中，选择是否允许 Cognito 自动发送消息以进行验证和确认。启用此设置后，Amazon Cognito 会向新用户发送电子邮件或短信，其中包含他们必须向您的用户池出示的代码。这将确认他们对电子邮件地址或电话号码的所有权，将等效属性设置为已验证，并确认用于登录的用户账户。您选择的要验证的属性将决定验证消息的传送方式和目的地。
2. 在创建用户时，验证属性更改并不重要，但与属性验证有关。可以允许已更改但尚未验证其登录属性的用户继续使用其新属性值或原始属性值登录。有关更多信息，请参阅 当用户更改其电子邮件或电话号码时应进行验证。
3. 必填属性显示用户注册或您创建用户之前必须为其提供值的属性。只有在创建用户池时才能设置必需的属性。
4. 自定义属性对用户创建和注册过程很重要，因为只有在首次创建用户时才能为不可变的自定义属性设置值。有关自定义属性的更多信息，请参阅自定义属性。
5. 如果您希望用户能够使用未经身份验证的 `SignUp` API 生成新账户，请在自助注册中，选择启用自行注册。如果您禁用自助注册，则只能在 Amazon Cognito 控制台中或通过 API 请求AdminCreateUser以管理员身份创建新用户。在自助注册处于非活动状态的用户池中，SignUpAPI 请求会返回`NotAuthorizedException`，托管登录不会显示注册链接。

对于计划以管理员身份创建用户的用户池，可以在登录菜单的设置中配置临时密码的持续时间。管理员设置的临时密码到期时间。

以管理员身份创建用户的另一个重要元素是邀请消息。创建新用户时，Amazon Cognito 会给他们发送一条包含您的应用程序链接的消息，以便他们可以首次登录。在 “消息模板” 下的 “身份验证方法” 菜单中自定义此消息模板。

您可以使用客户端密钥配置机密应用程序客户端（通常是 Web 应用程序），以防止在没有应用程序客户端密钥的情况下注册。作为安全最佳实践，请勿在公共应用程序客户端（通常是移动应用程序）中分发应用程序客户端密钥。您可以在 Amazon Cognito 控制台的应用程序客户端菜单中使用客户端密钥创建应用程序客户端。

Amazon Cognito user pools API

您可以通过编程方式在CreateUserPool或 UpdateUserPoolAPI 请求中设置用于在用户池中创建用户的参数。

该AdminCreateUserConfig元素为用户池的以下属性设置值。

1. 启用自助注册
2. 向管理员创建的新用户发送的邀请消息

如果将以下示例添加到完整的 API 请求正文中，可设置一个用户池（自助注册为非活动状态）和一封基本的邀请电子邮件。

```
"AdminCreateUserConfig": {
      "AllowAdminCreateUserOnly": true,
      "InviteMessageTemplate": {
         "EmailMessage": "Your username is {username} and temporary password is
 {####}.",
         "EmailSubject": "Welcome to ExampleApp",
         "SMSMessage": "Your username is {username} and temporary password is
 {####}."
      }
   }
```

CreateUserPool或 UpdateUserPoolAPI 请求的以下附加参数控制新用户的创建。

AutoVerifiedAttributes

注册新用户时要自动向其发送消息的属性、电子邮件地址或电话号码。

策略

用户池密码策略。

架构

用户池自定义属性。它们对用户的创建和注册过程很重要，因为只有在首次创建用户时才能为不可变的自定义属性设置值。

此参数还将为您的用户池设置必填属性。如果将以下文本插入到完整 API 请求正文的 `Schema` 元素中，将把 `email` 属性设为必填属性。

```
{
            "Name": "email",
            "Required": true
}
```

注册并确认用户账户

可通过以下任一方法将用户账户添加到您的用户池中：

- 用户在您用户池的客户端应用程序中进行注册。这可以是移动应用程序或 Web 应用程序。
- 您可以将用户账户导入到用户池中。有关更多信息，请参阅 通过 CSV 文件将用户导入用户池中。
- 您可以在用户池中创建用户账户并邀请用户登录。有关更多信息，请参阅 以管理员身份创建用户账户。

自行注册的用户必须得到确认才可登录。导入和创建的用户已经过确认，但他们必须在首次登录时创建自己的密码。以下部分将介绍确认过程以及电子邮件和电话验证。

注册时的密码

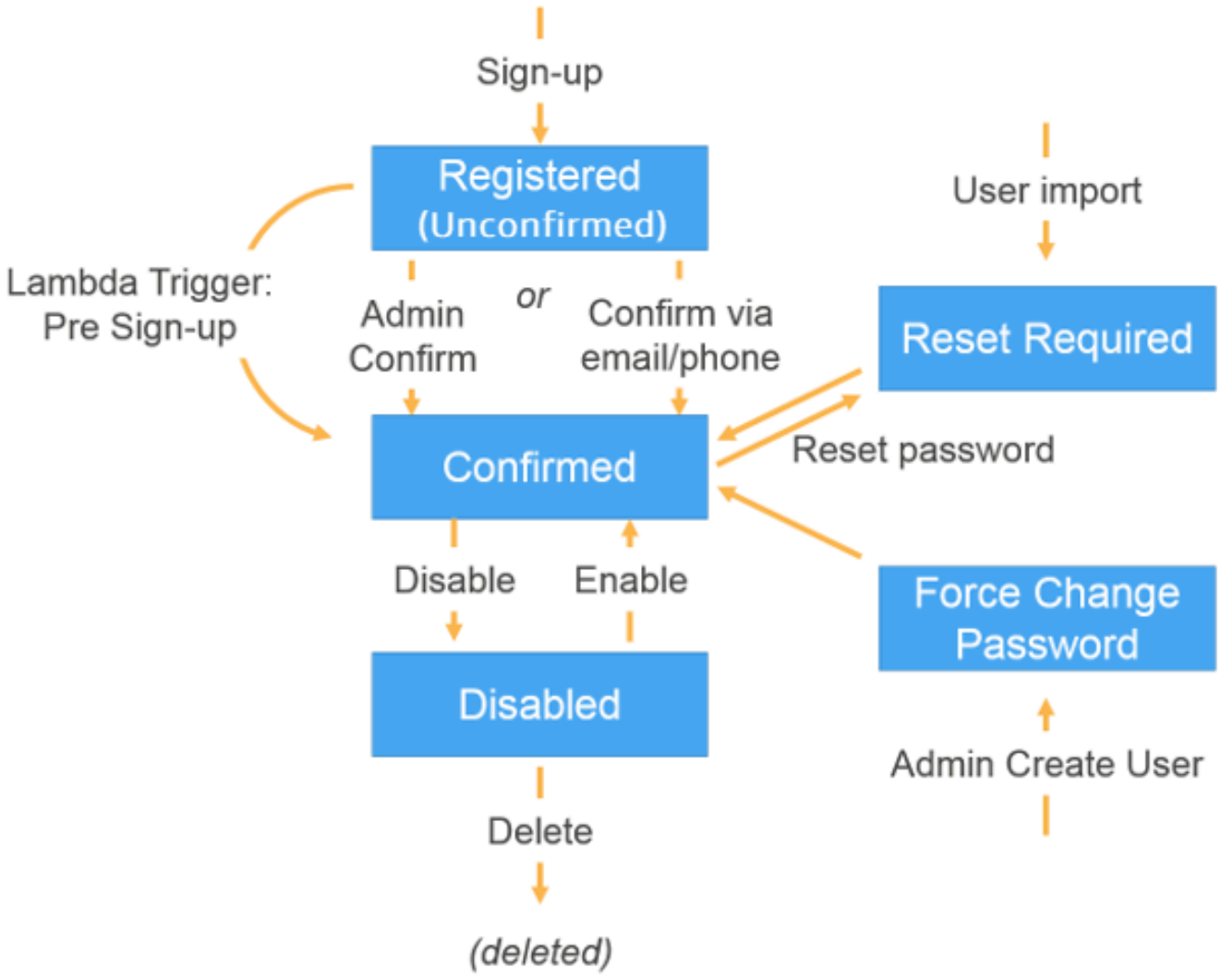
Amazon Cognito 要求所有用户在注册时提供密码，但以下情况除外。如果满足所有这些条件，则可以在注册操作中省略密码。

1. 无密码登录在您的用户池和应用程序客户端中处于活动状态。
2. 您的应用程序是使用 Amazon SDK 中的身份验证模块定制的。托管登录和托管用户界面始终需要密码。

3. 用户为您允许的无密码登录方法（电子邮件或短信一次性密码 ()OTPs）提供属性值。例如，如果您允许使用电子邮件和电话OTP登录，则用户可以提供电话号码或电子邮件地址，但是如果您只允许使用电子邮件登录，则他们必须提供电子邮件地址。
4. 您的用户池会自动验证用户可在无密码登录中使用的属性。
5. 对于任何给定的SignUp请求，用户都不会为 P asswor d 参数提供值。

用户账户确认概览

下图阐明了确认过程：



用户账户可以处于以下任一状态：

已注册（未确认）

用户已成功注册，但在用户账户得到确认之前无法登录。在此状态下，用户已启用，但未得到确认。

自行注册的新用户由此状态开始。

已确认

用户账户已确认，用户可以登录。当用户输入代码或点击电子邮件链接以确认其用户账户时，系统将自动验证电子邮件或电话号码。代码或链接的有效期为 24 小时。

如果管理员或预注册 Lambda 触发器确认了用户账户，则可能没有与该账户关联的经验证的电子邮件或电话号码。

需要重置密码

用户账户已确认，但用户必须请求代码并重置其密码，然后才可以登录。

由管理员或开发人员导入的用户账户以此状态开始。

强制更改密码

用户账户已确认，用户可以使用临时密码进行登录。但在首次登录时，用户必须将其密码更改为新值，然后才能执行任何其他操作。

由管理员或开发人员创建的用户账户以此状态开始。

已禁用

在可以删除用户账户之前，必须禁用该用户的登录访问权限。

更多资源

- 使用 Amazon Cognito 检测和修复不活跃的用户账户

在注册时验证联系人信息

当新用户注册您的应用程序时，您可能希望他们提供至少一种联系方式。例如，利用用户的联系人信息，您可以：

- 在用户选择重置其密码时发送临时密码。

- 在更新用户的个人信息或财务信息后向用户发送通知。
- 发送促销信息（例如，特别优惠或折扣）。
- 发送账户摘要或账单提醒。

对于像这样的使用案例，将消息发送到经过验证的目的地非常重要。否则，您可能会将消息发送到错误键入的无效电子邮件地址或电话号码。或者更糟糕的是，您可能会将敏感信息发送给冒充您的用户的坏人。

为了帮助确保您仅将消息发送给正确的人员，请配置您的 Amazon Cognito 用户池以使用户在注册时必须提供以下内容：

a. 一个电子邮件地址或电话号码。
b. Amazon Cognito 发送到该电子邮件地址或电话号码的验证码。如果 24 小时过去并且您的用户的代码或链接不再有效，请调用 ResendConfirmationCodeAPI 操作生成并发送新的代码或链接。

通过提供验证码，用户可以证明他们有权访问收到该代码的邮箱或手机。在用户提供该代码后，Amazon Cognito 将通过以下方式更新用户池中的用户相关信息：

- 将用户的状态设置为 `CONFIRMED`。
- 更新用户的属性以指示已验证电子邮件地址或电话号码。

要查看此信息，您可以使用 Amazon Cognito 控制台。或者，您可以使用 `AdminGetUser` API 操作、带的`admin-get-user` Amazon CLI命令或其中一个中的相应操作 Amazon SDKs。

如果用户具有经验证的联系方式，Amazon Cognito 会在用户请求重置密码时自动向其发送消息。

确认和验证用户属性的其他操作

以下用户活动验证了用户属性。您无需将这些属性设置为自动验证：列出的操作在所有情况下都将其标记为已验证。

电子邮件地址

1. 使用电子邮件一次性密码 (OTP) 成功完成无密码身份验证。
2. 使用电子邮件 OTP 成功完成多因素身份验证 (MFA)。

电话号码

1. 成功使用短信 OTP 完成无密码身份验证。

2. 使用短信 OTP 成功完成 MFA。

配置您的用户池以要求电子邮件或手机验证

验证用户的电子邮件地址和电话号码时，确保可以联系到用户。完成中的以下步骤， Amazon Web Services Management Console 将您的用户池配置为要求您的用户确认其电子邮件地址或电话号码。

> **Note**
>
> 如果您的账户中还没有用户池，请参阅用户池入门。

配置用户池

1. 导航到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 从导航窗格中选择用户池。从列表中选择一个现有用户池，或创建一个用户池。
3. 选择注册菜单并找到属性验证和用户帐户确认。选择编辑。
4. 在 Cognito 辅助验证和确认下，选择是否允许 Cognito 自动发送消息以进行验证和确认。启用此设置后，当用户注册或您创建用户配置文件时，Amazon Cognito 会向您选择的用户联系人属性发送消息。为验证属性并确认登录的用户配置文件，Amazon Cognito 会通过消息向用户发送代码或链接。用户随后必须在您的 UI 中输入相应代码，这样，您的应用才能在 `ConfirmSignUp` 或 `AdminConfirmSignUp` API 请求中对其进行确认。

   > **Note**
   >
   > 您还可以禁用 Cognito 辅助验证和确认并使用经过身份验证的 API 操作或 Lambda 触发器验证属性并确认用户。
   > 如果您选择此选项，则 Amazon Cognito 不会在用户注册时发送验证码。如果您要使用自定义身份验证流程验证至少一种联系方式，而不使用来自 Amazon Cognito 的验证码，请选择此选项。例如，您可以使用一个预注册 Lambda 触发器，该触发器将自动验证属于特定域的电子邮件地址。
   > 如果您不验证用户的联系人信息，用户可能无法使用您的应用程序。请记住，用户需要经验证的联系人信息才能：
   >
   > - 重置密码 – 当用户在您的应用程序中选择调用 `ForgotPassword` API 操作的选项时，Amazon Cognito 会将临时密码发送到用户的电子邮件地址或电话号码。Amazon Cognito 仅在用户具有至少一个经验证的联系方式时发送此密码。

- 通过使用电子邮箱地址或电话号码作为别名进行登录 – 如果您将用户池配置为允许这些别名，则用户只能在别名经过验证后使用别名进行登录。有关更多信息，请参阅 自定义登录属性。

5. 选择您的要验证的属性：

   发送 SMS 消息，验证电话号码

   Amazon Cognito 将在用户注册时通过 SMS 消息发送验证码。如果您通常通过 SMS 消息与用户通信，请选择此选项。例如，如果您要发送交付通知、约会确认或提醒，则将需要使用经过验证的电话号码。确认账户时，用户电话号码将成为已验证属性；您必须采取其它操作来验证用户电子邮件地址并与其进行通信。

   发送电子邮件消息，验证电子邮件地址

   Amazon Cognito 将在用户注册时通过电子邮件发送验证码。如果您通常通过电子邮件与用户通信，请选择此选项。例如，如果您要发送账单、订单摘要或特别优惠，则将需要使用经过验证的电子邮件地址。确认账户时，用户电子邮件地址将成为已验证属性；您必须采取其它操作来验证用户电话号码并与其进行通信。

   如果电话号码可用，则发送 SMS 消息，否则发送电子邮件

   如果您不要求所有用户都拥有相同的经验证的联系方式，请选择此选项。在这种情况下，您的应用程序中的注册页面可能会要求用户仅验证其首选联系方式。当 Amazon Cognito 发送验证码时，会将该代码发送到来自您应用程序的 `SignUp` 请求中提供的联系方式。如果用户同时提供了电子邮件地址和电话号码，并且您的应用程序在 `SignUp` 请求中提供了这两种联系方式，Amazon Cognito 将仅向电话号码发送验证码。

   如果您要求用户同时验证电子邮件地址和电话号码，则选择此选项。Amazon Cognito 将在用户注册时验证一种联系方式，而且您的应用程序必须在用户登录后验证另一种联系方式。有关更多信息，请参阅 在您要求用户确认电子邮件地址和电话号码的情况下。

6. 选择保存更改。

使用电子邮件或电话验证的身份验证流程

如果您的用户池要求用户验证其联系人信息，则当用户注册时，您的应用程序必须促进以下流程：

1. 用户通过输入用户名称、电话号码和/或电子邮件地址及其他可能属性，在您的应用程序中进行注册。

2. Amazon Cognito 服务接收来自应用程序的注册请求。验证该请求包含注册所需的所有属性后，该服务将完成注册过程并向用户的手机（通过 SMS 消息）或电子邮件发送确认码。该代码的有效期为 24 小时。
3. 该服务向应用程序返回信息，表示注册过程已完成且用户账户正等待确认。响应中包含关于确认代码所发送到位置的信息。此时，用户账户处于未确认状态，而且用户的电子邮件地址和电话号码未经验证。
4. 现在，应用程序会提示用户输入确认代码。用户无需立即输入代码。但是，用户只有在输入确认代码后才可登录。
5. 用户在应用程序中输入确认代码。
6. 应用程序调用 `ConfirmSignUp` 将代码发送到 Amazon Cognito 服务，该服务将验证代码并在代码正确时将用户账户设置为已确认状态。成功确认用户账户之后，Amazon Cognito 服务会自动将用于确认（电子邮件地址或电话号码）的属性标记为已验证。除非此属性的值发生更改，否则用户无需再次进行验证。
7. 此时，用户账户处于已确认状态，用户可以登录。

在您要求用户确认电子邮件地址和电话号码的情况下

Amazon Cognito 在用户注册时仅验证一种联系方式。如果 Amazon Cognito 必须在验证电子邮件地址或电话号码之间进行选择，则会选择通过 SMS 消息发送验证码来验证电话号码。例如，如果您将用户池配置为允许用户验证电子邮件地址或电话号码，并且您的应用程序在注册时提供了这两个属性，则 Amazon Cognito 将仅验证电话号码。在用户验证其电话号码后，Amazon Cognito 会将用户的状态设置为 `CONFIRMED`，并允许用户登录您的应用程序。

在用户登录后，您的应用程序会提供相应选项来验证在注册期间未验证的联系方式。为了验证第二种联系方式，您的应用程序将调用 `VerifyUserAttribute` API 操作。请注意，此操作需要 `AccessToken` 参数，而 Amazon Cognito 只为经过身份验证的用户提供访问令牌。因此，您只能在用户登录后验证第二种联系方式。

如果您要求用户同时验证电子邮件地址和电话号码，请执行以下操作：

1. 配置用户池以允许用户验证电子邮件地址或电话号码。
2. 在应用程序的注册流程中，要求用户提供电子邮件地址和电话号码。调用 `SignUp` API 操作，并为 `UserAttributes` 参数提供电子邮件地址和电话号码。此时，Amazon Cognito 会向用户的手机发送一个验证码。

3. 在应用程序界面中，会显示一个确认页面以供用户输入验证码。通过调用 ConfirmSignUp API 操作来确认用户。此时，用户的状态为 CONFIRMED，并且用户的电话号码已验证，但电子邮件地址未验证。
4. 显示登录页，并通过调用 InitiateAuth API 操作对用户进行身份验证。对用户进行身份验证后，Amazon Cognito 将向您的应用程序返回访问令牌。
5. 调用 GetUserAttributeVerificationCode API 操作。在请求中指定以下参数：

   - AccessToken – Amazon Cognito 在用户登录时返回的访问令牌。
   - AttributeName – 将 "email" 指定为属性值。

   Amazon Cognito 将向用户的电子邮件地址发送一个验证码。
6. 显示一个确认页面以供用户输入验证码。当用户提交代码时，请调用 VerifyUserAttribute API 操作。在请求中指定以下参数：

   - AccessToken – Amazon Cognito 在用户登录时返回的访问令牌。
   - AttributeName – 将 "email" 指定为属性值。
   - Code – 用户提供的验证码。

   此时，电子邮件地址已验证。

允许用户在您的应用程序中注册但以用户池管理员身份进行确认

您可能不希望您的用户池自动在用户池中发送验证消息，但仍希望允许任何人注册账户。例如，该模型为人工审查新的注册请求以及批量验证和处理注册留出了空间。您可以在 Amazon Cognito 控制台中或通过 IAM 身份验证的 API 操作确认新的用户账户。AdminConfirmSignUp无论您的用户池是否发送验证消息，您都能以管理员身份确认用户账户。

您只能使用此方法确认用户的自助注册。要确认您以管理员身份创建的用户，请创建Permanent设置为AdminSetUserPassword的 API 请求True。

1. 用户通过输入用户名称、电话号码和/或电子邮件地址及其他可能属性，在您的应用程序中进行注册。
2. Amazon Cognito 服务接收来自应用程序的注册请求。验证该请求包含注册所需的所有属性之后，该服务将完成注册过程，并向应用程序返回信息，表示注册已完成且正在等待确认。此时，用户账户处于未确认状态。账户经过确认后，用户才可登录。

3. 确认用户的账户。您必须使用 Amazon 凭据登录 Amazon Web Services Management Console 或签署 API 请求才能确认账户。

   a. 要在 Amazon Cognito 控制台中确认用户，请导航至 “用户” 菜单，选择要确认的用户，然后从 “操作” 菜单中选择 “确认”。

   b. 要在 Amazon API 或 CLI 中确认用户，请创建 AdminConfirmSignUpAPI 请求，或者admin-confirm-sign-up在 Amazon CLI。

4. 此时，用户账户处于已确认状态，用户可以登录。

计算密钥哈希值

作为最佳实践，请将客户端密钥分配给您的机密应用程序客户端。当您向应用程序客户端分配客户端密钥时，您的 Amazon Cognito 用户池 API 请求必须包括一个哈希值，用于包含请求正文中的客户端密钥。为了验证您对以下列表中 API 操作的客户端密钥的了解，请将客户端密钥与应用程序客户端 ID 和用户的用户名连接起来，然后对该字符串进行 base64 编码。

应用程序将用户登录到具有密钥哈希值的客户端时，您可以使用任何用户池登录属性的值作为密钥哈希值的用户名元素。应用程序在使用 `REFRESH_TOKEN_AUTH` 的身份验证操作中请求新令牌时，用户名元素的值取决于您的登录属性。如果您的用户池没有将 `username` 用作登录属性，请通过用户的访问令牌或 ID 令牌中的 `sub` 声明设置密钥哈希用户名值。如果 `username` 为登录属性，请通过 `username` 声明设置密钥哈希用户名值。

以下 Amazon Cognito 用户池 APIs 接受参数中的客户端密钥哈希值。`SecretHash`

- ConfirmForgotPassword
- ConfirmSignUp
- ForgotPassword
- ResendConfirmationCode
- SignUp

此外，以下内容 APIs 接受`SECRET_HASH`参数中的客户端密钥哈希值，无论是在身份验证参数中还是在质询响应中。

API 操作	的父参数 SECRET_HASH
InitiateAuth	AuthParameters

API 操作	的父参数 SECRET_HASH
AdminInitiateAuth	AuthParameters
RespondToAuthChallenge	ChallengeResponses
AdminRespondToAuthChallenge	ChallengeResponses

密钥哈希值是 Base 64 编码的加密哈希消息身份验证代码（HMAC，Hash Message Authentication Code），使用用户池客户端的私有密钥、用户名以及消息中的客户端 ID 进行计算。以下伪代码显示如何计算此值。在此伪代码中，+表示串联，表示使用 Hmac 生成 HMAC 值的函数，并`HMAC_SHA256``Base64`表示生成 Base-64 编码`SHA256`版本的哈希输出的函数。

```
Base64 ( HMAC_SHA256 ( "Client Secret Key", "Username" + "Client Id" ) )
```

有关如何计算和使用`SecretHash`参数的详细概述，请参阅如何解决我的 Amazon Cognito 用户池 API 中的 “无法验证客户端的秘密哈希” 错误？<client-id> 在 Amazon 知识中心中。

您可以在服务器端应用程序代码中使用以下代码示例：

Shell

```
echo -n "[username][app client ID]" | openssl dgst -sha256 -hmac [app client secret]
 -binary | openssl enc -base64
```

Java

```
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;

public static String calculateSecretHash(String userPoolClientId, String
 userPoolClientSecret, String userName) {
    final String HMAC_SHA256_ALGORITHM = "HmacSHA256";

    SecretKeySpec signingKey = new SecretKeySpec(
            userPoolClientSecret.getBytes(StandardCharsets.UTF_8),
            HMAC_SHA256_ALGORITHM);
    try {
        Mac mac = Mac.getInstance(HMAC_SHA256_ALGORITHM);
        mac.init(signingKey);
```

```
        mac.update(userName.getBytes(StandardCharsets.UTF_8));
        byte[] rawHmac =
 mac.doFinal(userPoolClientId.getBytes(StandardCharsets.UTF_8));
        return Base64.getEncoder().encodeToString(rawHmac);
    } catch (Exception e) {
        throw new RuntimeException("Error while calculating ");
    }
}
```

Python

```
import sys
import hmac, hashlib, base64
username = sys.argv[1]
app_client_id = sys.argv[2]
key = sys.argv[3]
message = bytes(sys.argv[1]+sys.argv[2],'utf-8')
key = bytes(sys.argv[3],'utf-8')
secret_hash = base64.b64encode(hmac.new(key, message,
 digestmod=hashlib.sha256).digest()).decode()
print("SECRET HASH:",secret_hash)
```

无需验证电子邮件或电话号码即可确认用户账户

预注册 Lambda 触发器可用于在注册时自动确认用户账户，而无需提供确认码或者验证电子邮件或电话号码。通过此方法进行确认的用户可立即登录，而无需接收代码。

您还可通过此触发器将用户的电子邮件或电话号码标记为已验证。

> **Note**
>
> 虽然这种方法对刚入门的用户而言很方便，但我们建议至少自动验证电子邮件或电话号码之一。否则，如果用户忘记密码，可能就无法进行恢复。

如果您不要求用户在注册时接收并输入确认码，也不在预注册 Lambda 触发器中自动验证电子邮件和电话号码，则您将承担对于该用户账户不具备经验证的电子邮件地址或电话号码的风险。用户可以稍后验证电子邮件地址或电话号码。但是，如果用户忘记自己的密码且没有经验证的电子邮件地址或电话号码，则用户将被锁定而无法使用账户，因为忘记密码流程需要经验证的电子邮件或电话号码以便向用户发送验证码。

当用户更改其电子邮件或电话号码时应进行验证

在您配置了多个登录名的用户池中，用户可以在登录时输入电话号码或电子邮件地址作为用户名。当他们在您的应用程序中更新其电子邮件地址或电话号码时，Amazon Cognito 可以立即向他们发送一条消息，其中包含验证他们对新属性值的所有权的代码。要启用自动发送这些验证码，请参阅配置电子邮件或电话验证。

收到验证码的用户必须在请求中将该验证码提供给 Amazon Cognito。VerifyUserAttribute在他们提供代码后，他们的属性将被标记为已验证。通常，当用户更新其电子邮件地址或电话号码时，您需要先验证他们是否拥有新值，然后他们才能使用新值登录和接收消息。用户池有一个可配置的选项，用于确定用户是否必须验证其电子邮件地址或电话号码的更新。

此选项是用户池属性`AttributesRequireVerificationBeforeUpdate`。在CreateUserPool或UpdateUserPool请求中对其进行配置，或者在 Amazon Cognito 控制台的 “注册” 菜单中使用待更新时保持原始属性值处于活动状态设置。

您的用户池如何处理电子邮件地址和电话号码的更新与用户池的用户名配置有关。用户池用户名可以采用用户名属性配置，其中登录名为电子邮件地址、电话号码或两者兼而有之。它们也可以位于别名属性配置中，其中`username`属性是登录名以及电子邮件地址、电话号码或首选用户名作为备用登录名。有关更多信息，请参阅 自定义登录属性。

您也可以使用自定义消息 Lambda 触发器来自定义验证消息。有关更多信息，请参阅 自定义消息 Lambda 触发器。当用户的电子邮件地址或电话号码未经验证时，您的应用程序应通知用户他们必须验证该属性，并提供一个按钮或链接供用户输入其验证码。

下表描述了当用户更改其登录属性的值时，别名设置如何`AttributesRequireVerificationBeforeUpdate`决定结果。

用户名配置	用户必须验证新属性时的行为	不需要用户验证新属性时的行为
用户名属性	原始属性仍处于验证状态，有资格登录，且为原始值。当用户验证新值时，Amazon Cognito 会更新属性值，将其标记为已验证，并使其符合登录资格。	亚马逊 Cognito 将属性更新为新值。新值有资格登录。当用户验证新值时，Amazon Cognito 会将其标记为已验证。
别名属性	原始属性仍处于验证状态，有资格登录，且为原始值。	亚马逊 Cognito 将属性更新为新值。原始属性值或新属性值

用户名配置	用户必须验证新属性时的行为	不需要用户验证新属性时的行为
	当用户验证新值时，Amazon Cognito 会更新属性值，将其标记为已验证，并使其符合登录资格。	均不符合登录资格。当用户验证新值时，Amazon Cognito 会更新属性值，将其标记为已验证，并使其符合登录资格。

示例 1

用户 1 使用电子邮件地址登录您的应用程序，`user1@example.com`并拥有用户名`user1`（别名属性）。您的用户池已配置为验证登录属性的更新并自动发送验证消息。他们要求将其电子邮件地址更新为`user1+foo@example.com`。他们会收到一封验证电子邮件，`user1+foo@example.com`并且只能使用该电子邮件地址重新登录`user1@example.com`。之后，他们输入验证码，并且只能使用该电子邮件地址再次登录`user1+foo@example.com`。

示例 2

用户 2 使用电子邮件地址登录您的应用程序，`user2@example.com`并拥有用户名（别名属性）。您的用户池配置为不验证登录属性的更新，而是自动发送验证消息。他们要求将其电子邮件地址更新为`user2+bar@example.com`。他们在收到一封验证电子邮件`user2+bar@example.com`，但无法再次登录。之后，他们输入验证码，并且只能使用该电子邮件地址再次登录`user2+bar@example.com`。

示例 3

用户 3 使用电子邮件地址登录您的应用程序`user3@example.com`，但没有用户名（用户名属性）。您的用户池配置为不验证登录属性的更新，而是自动发送验证消息。他们要求将其电子邮件地址更新为`user3+baz@example.com`。他们会收到一封验证电子邮件`user3+baz@example.com`，但他们可以立即登录，无需使用验证码进行任何其他操作。

针对由管理员或开发人员创建的用户账户的确认和验证过程

由管理员或开发人员创建的用户账户已经处于已确认状态，所以用户无需输入确认代码。Amazon Cognito 服务向这些用户发送的邀请消息包含用户名和临时密码。用户需要在登录前更改密码。有关更多信息，请参阅自定义电子邮件和 SMS 消息中的 以管理员身份创建用户账户和使用 Lambda 触发器自定义用户池工作流中的自定义消息触发器。

针对导入的用户账户的确认和验证过程

使用 Amazon Web Services Management Console、CLI 或 API 中的用户导入功能（参见通过 CSV 文件将用户导入用户池中）创建的用户账户已处于已确认状态，因此用户无需输入确认码。没有发送邀请消息。但是，导入的用户账户要求用户首先调用 `ForgotPassword` API 来请求代码，然后通过调用 `ConfirmForgotPassword` API 来使用发送的代码创建密码，之后方可登录。有关更多信息，请参阅要求导入的用户重置密码。

导入用户账户时，用户的电子邮件或电话号码必须已标记为已验证，从而用户无需验证即可登录。

在测试应用程序时发送电子邮件

当用户在您的用户池的客户端应用程序中创建和管理其账户时，Amazon Cognito 将向用户发送电子邮件。如果您将用户池配置为要求电子邮件验证，Amazon Cognito 将在以下情况下发送电子邮件：

- 用户注册。
- 用户更新其电子邮件地址。
- 用户执行一项调用 `ForgotPassword` API 操作的操作。
- 您以管理员身份创建用户账户。

根据发起电子邮件递送的操作，电子邮件将包含验证码或临时密码。您的用户必须接收这些电子邮件并理解消息。否则，他们可能无法登录并使用您的应用程序。

要确保电子邮件成功发送并且邮件看起来正确，请测试应用程序中从 Amazon Cognito 启动电子邮件传送的操作。例如，通过使用应用程序中的注册页面或通过使用 `SignUp` API 操作，您可以通过使用测试电子邮件地址进行注册来启动电子邮件传送。在通过此方式进行测试时，请记住以下几点：

> **重要提示**
>
> 当您使用电子邮件地址测试从 Amazon Cognito 启动电子邮件传送的操作时，请勿使用虚假的电子邮件地址（没有邮箱的电子邮件地址）。使用真实的电子邮件地址才能接收来自 Amazon Cognito 的电子邮件，而不会创建查无此人的邮件。
> 在 Amazon Cognito 未能将电子邮件传送到收件人邮箱时会产生查无此人的邮件，如果邮箱不存在，则始终会发生这种情况。
> Amazon Cognito 限制了持续出现硬退邮件的 Amazon 账户可以发送的电子邮件数量。

当您测试启动电子邮件传送的操作时，请使用下列电子邮件地址之一以防止出现查无此人的邮件：

- 您拥有的用于测试的电子邮件账户的地址。当您使用自己的电子邮件地址时，您将收到 Amazon Cognito 发送的电子邮件。利用此电子邮件，您可以使用验证码来测试应用程序中的注册体验。如果您为用户池自定义了电子邮件，则可检查自定义项看起来是否正确。
- 邮箱模拟器地址：success@simulator.amazonses.com。如果您使用模拟器地址，Amazon Cognito 将成功发送电子邮件，但您无法查看。当您不需要使用验证码并且不需要检查电子邮件时，此选项很有用。
- 添加了任意标签的邮箱模拟器地址（如 success+user1@simulator.amazonses.com 或 success+user2@simulator.amazonses.com）。Amazon Cognito 可成功向这些地址发送电子邮件，但您无法查看其发送的电子邮件。当您希望通过向用户池添加多个测试用户来测试注册过程，并且每个测试用户都具有一个唯一的电子邮件地址时，此选项很有用。

配置电子邮件或电话验证

您可以在 “身份验证方法” 菜单下选择电子邮件或电话验证的设置。有关多重验证 (MFA) 的详细信息，请参阅SMS 文本消息 MFA。

Amazon Cognito 使用 Amazon SNS 发送 SMS 消息。如果您 Amazon Web Services 服务 之前没有发送过来自亚马逊 Cognito 或其他任何公司的短信，Amazon SNS 可能会将您的账户置于短信沙箱中。我们建议您在将账户从沙盒移到生产环境之前，向已验证的电话号码发送测试消息。此外，如果您计划向美国的目标电话号码发送短信，则必须从 Amazon Pinpoint 获取源 ID 或发件人 ID。要为 Amazon Cognito 用户群体配置 SMS 消息，请参阅 Amazon Cognito 用户池的短信设置。

Amazon Cognito 可以自动验证电子邮件地址或电话号码。要进行此验证，Amazon Cognito 将发送验证码或验证链接。对于电子邮件地址，Amazon Cognito 可以通过电子邮件发送代码或链接。在 Amazon C ognito 控制台的消息模板菜单中编辑验证消息模板时，您可以选择验证码或链接的验证类型。有关更多信息，请参阅 自定义电子邮件验证消息。

对于电话号码，Amazon Cognito 以 SMS 文本消息的形式发送代码。

Amazon Cognito 必须验证电话号码或电子邮件地址来确认用户，帮助他们恢复忘记的密码。或者，您可以使用注册前 Lambda 触发器或使用 API 操作自动确认用户AdminConfirmSignUp。有关更多信息，请参阅 注册并确认用户账户。

验证代码或链接的有效期为 24 小时。

如果您选择要求通过电子邮件地址或电话号码进行验证，则在用户注册时，Amazon Cognito 将自动发送验证代码或链接。如果用户池已配置了 自定义 SMS 发件人 Lambda 触发器 或 自定义电子邮件发件人 Lambda 触发器，则会改为调用该函数。

> **备注**
>
> - Amazon SNS 会另外收取用于验证电话号码的 SMS 文本消息费用。发送电子邮件不收费。有关 Amazon SNS 定价的信息，请参阅 Worldwide SMS 定价。有关提供 SMS 消息收发服务的最新国家/地区列表，请参阅支持的区域和国家/地区。
> - 当您在应用程序中测试从 Amazon Cognito 生成电子邮件的操作时，请使用 Amazon Cognito 可以发送到而不会查无此人的邮件的真实电子邮件地址。有关更多信息，请参阅 the section called “在测试应用程序时发送电子邮件”。
> - 忘记密码流程要求通过用户的电子邮件或电话号码来验证用户。

> **Important**
>
> 如果用户同时注册了电话号码和电子邮件地址，并且用户群体设置需要验证这两个属性，那么 Amazon Cognito 会通过 SMS 消息将验证码发送到电话号码上。Amazon Cognito 尚未验证电子邮件地址，因此您的应用程序必须致电GetUser以查看电子邮件地址是否在等待验证。如果确实需要验证，则应用程序必须致电GetUserAttributeVerificationCode以启动电子邮件验证流程。然后它必须通过致电提交验证码VerifyUserAttribute。

您可以调整 Amazon Web Services 账户 和单条消息的短信支出配额。该限额仅适用于发送 SMS 消息的费用。有关更多信息，请参阅什么是账户级和消息级支出配额及其运作方式？ 在 Amazon SNS FAQs 中。

Amazon Cognito 使用您创建用户池的地方或下表中传统的 Amazon SNS 备用区域的 Amazon SNS 资源发送短信。 Amazon Web Services 区域 亚太地区（首尔）区域中的 Amazon Cognito 用户池例外。这些用户池使用您在亚太地区（东京）区域中的 Amazon SNS 配置。有关更多信息，请参阅 选择 Amazon SNS 短信 Amazon Web Services 区域。

Amazon Cognito 区域	旧版 Amazon SNS 备用区域
美国东部（俄亥俄州）	美国东部（弗吉尼亚州北部）
亚太地区（孟买）	亚太地区（新加坡）
亚太地区（首尔）	亚太地区（东京）

Amazon Cognito 区域	旧版 Amazon SNS 备用区域
加拿大（中部）	美国东部（弗吉尼亚州北部）
欧洲地区（法兰克福）	欧洲地区（爱尔兰）
欧洲地区（伦敦）	欧洲地区（爱尔兰）

示例：如果您的 Amazon Cognito 用户池位于亚太地区（孟买）区域，并且您增加了在 ap-southeast-1 区域中的支出限额，则可能不希望请求单独增加 ap-south-1 的支出限额。而是可以使用亚太地区（新加坡）中的 Amazon SNS 资源。

验证对于电子邮件地址和电话号码的更新

在用户更改电子邮件地址或电话号码属性的值后，这些属性可以立即变为活动但未经验证的状态。Amazon Cognito 还可以要求您的用户在 Amazon Cognito 更新属性之前验证新值。当您要求用户首先验证新值时，他们可以使用原始值进行登录和接收消息，直到他们验证新值。

当您的用户可以使用其电子邮件地址或电话号码作为用户群体中的登录别名时，他们的已更新属性的登录名取决于您是否要求验证已更新的属性。当您要求用户验证已更新的属性时，用户可以使用原始属性值登录，直到他们验证新值。如果您不要求用户验证已更新的属性，则在验证新值之前，用户无法使用新属性值或原始属性值登录或接收消息。

例如，您的用户群体允许使用电子邮件地址别名登录，并要求用户在更新时验证其电子邮件地址。Sue 以 `sue@example.com` 身份登录，想将她的电子邮件地址更改为 `sue2@example.com`，但是不小心输入了 **s**`sue2@example.com`。Sue 没有收到验证电子邮件，所以她无法验证 `ssue2@example.com`。Sue 以 `sue@example.com` 身份登录，然后在您的应用程序中重新提交表单，以将她的电子邮件地址更新为 `sue2@example.com`。她收到此电子邮件，向您的应用程序提供验证码，然后开始以 `sue2@example.com` 身份登录。

当用户更新属性并且您的用户群体验证新的属性值时

- 在确认代码以验证新值之前，用户可以使用原始属性值登录。
- 用户只有在确认代码以验证新值后，才能使用新属性值登录。
- 如果您在 AdminUpdateUserAttributesAPI 请求`true`中`phone_number_verified`将`email_verified`或设置为，则他们可以在确认 Amazon Cognito 发送给他们的代码之前登录。

当用户更新属性而您的用户群体未验证新的属性值时

- 用户无法使用原始属性值登录或接收消息。
- 在确认代码以验证新属性值之前，用户无法使用新属性值登录或接收除确认码之外的消息。
- 如果您在 AdminUpdateUserAttributesAPI 请求`true`中`phone_number_verified`将`email_verified`或设置为，则他们可以在确认 Amazon Cognito 发送给他们的代码之前登录。

当用户更新其电子邮件地址或电话号码时，需要进行属性验证

1. 登录 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 在 “注册” 菜单中，选择 “属性验证和用户帐户确认” 下的 “编辑”。
4. 选择 Keep original attribute value active when an update is pending（等待更新时，保持原始属性值处于活动状态）。
5. 在 Active attribute values when an update is pending（等待更新时的活动属性值）下，选择您希望在 Amazon Cognito 更新值之前要求用户验证的属性。
6. 选择 Save changes（保存更改）。

要要求使用 Amazon Cognito API 进行属性更新验证，您可以在请求中设置`AttributesRequireVerificationBeforeUpdate`参数。UpdateUserPool

授权 Amazon Cognito 代表您发送消息。

要代表您向您的用户发送短信，Amazon Cognito 需要具有您的权限。要授予该权限，您可以创建一个 Amazon Identity and Access Management (IAM) 角色。在 Amazon Cognito 控制台的 “身份验证方法” 菜单中，选择 “短信” 下的 “编辑” 来设置角色。

配置验证和邀请消息

借助 Amazon Cognito，您可以自定义短信和电子邮件验证消息以及用户邀请消息，从而增强应用程序的安全性和用户体验。借助 Amazon Cognito，您可以在基于代码的验证或一键式链接验证之间进行选择，从而满足您的应用程序需求。本主题讨论如何在 Amazon Cognito 控制台中对多重身份验证（MFA）和验证通信进行个性化设置。

在消息模板菜单中，您可以自定义：

- 您的 SMS 短信多重身份验证（MFA）消息

- 您的 SMS 和电子邮件验证消息
- 电子邮件的验证类型–代码或链接

> **Note**
>
> 当用户注册或重新发送确认码时，Amazon Cognito 会在验证消息中发送包含您的基于链接的模板的链接。来自属性更新和密码重置操作的电子邮件使用代码模板。

- 您的用户邀请消息
- 流经用户池的电子邮件的 FROM 和 REPLY-TO 电子邮件地址

> **Note**
>
> 仅当您选择要求进行电话号码和电子邮件验证时，才会显示短信和电子邮件验证消息模板。同样，只有 MFA 设置为 required（必填）或 optional（可选）时，才会显示 SMS MFA 消息模板。

主题

- 消息模板
- 自定义 SMS 消息
- 自定义电子邮件验证消息
- 自定义用户邀请消息
- 自定义您的电子邮件地址
- 授权 Amazon Cognito 代表您发送 Amazon SES 电子邮件（通过自定义 FROM 电子邮件地址）

消息模板

您可以使用消息模板，在消息中插入占位符。Amazon Cognito 使用相应的值替换占位符。您可以在任何类型的消息模板中引用通用模板占位符，但并非所有消息类型中都会出现这些值。

通用模板占位符

描述	令牌	消息类型
验证代码	{####}	验证、确认和 MFA 消息

描述	令牌	消息类型
临时密码	{####}	忘记密码和邀请消息
用户名称	{username}	邀请和高级安全消息

具有威胁防护功能的可用自动响应之一是通知用户 Amazon Cognito 检测到潜在的恶意活动。您可以使用高级安全模板占位符执行以下操作：

- 包括某个事件的特定详细信息，例如 IP 地址、城市、国家/地区、登录时间、设备名称。Amazon Cognito 高级安全功能可以分析这些详细信息。
- 验证一键式链接是否有效。
- 使用事件 ID、反馈令牌和用户名构建您自己的一键式链接。

> **Note**
>
> 要生成一键式链接并在高级安全电子邮件模板中使用 {one-click-link-valid} 和 {one-click-link-invalid} 占位符，您必须已经为用户群体配置了域。

高级安全特征添加了以下占位符，您可以将其插入到消息模板中：

高级安全模板占位符

描述	令牌
IP 地址	{ip-address}
城市	{city}
Country	{country}
登录时间	{login-time}
设备名称	{device-name}
一键式链接有效	{one-click-link-valid}

描述	令牌
一键式链接无效	`{one-click-link-invalid}`
事件 ID	`{event-id}`
反馈令牌	`{feedback-token}`

自定义 SMS 消息

要自定义用于多重身份验证 (MFA) 的短信，请在 Amazon Cognito 用户池控制台的消息模板菜单中编辑 MFA 消息。

> **Important**
>
> 您的自定义消息必须包含 `{####}` 占位符。该占位符会在消息发送之前替换为身份验证代码。

Amazon Cognito 将包括身份验证码在内的短信的最大长度设置为 140 个 UTF-8 字符。

自定义 SMS 验证消息

要自定义用于电话号码验证的 SMS 消息，请在用户池的 “消息模板” 菜单中编辑验证消息模板。

> **Important**
>
> 您的自定义消息必须包含 `{####}` 占位符。该占位符会在消息发送之前替换为验证代码。

消息的最大长度为 140 个 UTF-8 字符，其中包括验证代码。

自定义电子邮件验证消息

要使用 Amazon Cognito 验证用户池中用户的电子邮件地址，您可以向用户发送一封电子邮件，其中包含用户可以点击的链接或可以输入的代码。

要自定义电子邮件地址验证消息的电子邮件主题和邮件内容，请在用户池的 “邮件模板” 菜单中编辑验证消息模板。当您编辑验证消息模板时，您可以选择验证类型，即代码或链接。

当您选择代码作为验证类型时，您的自定义消息必须包含 `{####}` 占位符。发送消息时，验证代码会替换占位符。

当您选择链接作为验证类型时，您的自定义消息必须包含格式为 {##Verify Your Email##} 的占位符。您可以更改占位符之间的文本字符串，例如 {##Click here##}。标题为 Verify Your Email（验证您的电子邮件）的验证链接将替换此占位符。

电子邮件验证消息的链接将您的用户定向到类似于以下示例的 URL。

```
https://<your user pool domain>/confirmUser/?
client_id=abcdefg12345678&user_name=emailtest&confirmation_code=123456
```

消息的最大长度为 20000 个 UTF-8 字符，其中包括验证代码（如果有）。您可以在此消息中使用 HTML 标签来格式化内容。

自定义用户邀请消息

您可以通过编辑消息模板菜单中的邀请消息模板来自定义 Amazon Cognito 通过短信或电子邮件向新用户发送的用户邀请消息。

> **Important**
>
> 您的自定义消息必须包含 {username} 和 {####} 占位符。当 Amazon Cognito 发送邀请消息时，它会将这些占位符替换为您用户的用户名和密码。

SMS 消息的最大长度为 140 个 UTF-8 字符，其中包括验证代码。电子邮件的最大长度为 20000 个 UTF-8 字符，其中包括验证代码。您可以在电子邮件中使用 HTML 标签来格式化内容。

自定义您的电子邮件地址

默认情况下，Amazon Cognito 通过 no-reply@verificationemail.com 向用户池中的用户发送电子邮件。您可以选择指定自定义 FROM 和 REPLY-TO 电子邮件地址来代替 no-reply@verificationemail.com。

自定义 FROM 和 REPLY-TO 电子邮件地址

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “身份验证方法” 菜单。在 Email（电子邮件）下，选择 Edit（编辑）。
4. 选择 SES Region（SES 区域）。
5. 从在您所选 SES Region（SES 区域）中已经过 Amazon SES 验证的电子邮件地址列表中，选择 FROM email address（发件人电子邮件地址）。要使用来自经过验证的域名的电子邮件地址，请

在 Amazon Command Line Interface 或 Amazon API 中配置电子邮件设置。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的在 Amazon SES 中验证电子邮件地址和域。

6. 从您所选 SES Region（SES 区域）中配置集的列表中，选择 Configuration set（配置集）。
7. 为您的电子邮件消息输入易记且格式为 `John Stiles <johnstiles@example.com>` 的 FROM sender name（FROM 发件人名称）。
8. 要自定义 REPLY-TO 电子邮件地址，请在 REPLY-TO email address（REPLY-TO 电子邮件地址）字段中输入有效的电子邮件地址。

授权 Amazon Cognito 代表您发送 Amazon SES 电子邮件（通过自定义 FROM 电子邮件地址）

您可以将 Amazon Cognito 配置为从自定义 FROM 电子邮件地址而不是默认地址发送电子邮件。要使用自定义地址，您必须授予 Amazon Cognito 权限，才能从经过 Amazon SES 验证的身份发送电子邮件。大多数情况下，您可以创建发送授权策略来授予权限。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的使用 Amazon SES 的发送授权。

当您将用户池配置为使用 Amazon SES 处理电子邮件时，Amazon Cognito 会在您的账户中创建 `AWSServiceRoleForAmazonCognitoIdpEmailService` 角色来授予对 Amazon SES 的访问权限。使用 `AWSServiceRoleForAmazonCognitoIdpEmailService` 服务相关角色时无需发送授权策略。只需在用户池中使用默认电子邮件功能和经过验证的 Amazon SES 身份作为 FROM 地址时，才需要添加发送授权策略。

有关 Amazon Cognito 创建的服务相关角色的更多信息，请参阅对 Amazon Cognito 使用服务相关角色。

以下示例发送授权策略授予 Amazon Cognito 使用经 Amazon SES 验证的身份的有限能力。Amazon Cognito 在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能发送电子邮件。有关更多示例，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略示例。

> **Note**
>
> 在此示例中，“Sid”值为唯一标识语句的任意字符串。有关策略语法的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略。

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

```
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

当您从下拉菜单中选择 Amazon SES 身份时，Amazon Cognito 控制台会为您添加相似策略。如果您使用 CLI 或 API 配置用户池，则必须将与前例结构相同的策略附加到您的 Amazon SES 身份。

以管理员身份创建用户账户

用户池不仅仅是客户身份和访问管理（CIAM）用户目录，互联网上的任何人都可以在您的应用程序中注册用户配置文件。您可以禁用自助注册。您可能已经认识您的客户，并且只想接纳那些事先获得授权的客户。您可以使用私有 SAML 2.0 或 OIDC 身份提供者、通过导入用户、通过在注册时筛选用户，或者通过使用管理 API 操作创建用户，您可以在应用程序周围设置手动身份验证护栏。您的用户管理创建工作流程可以是编程的，即在用户在其他系统中注册后对其进行配置，也可以在 Amazon Cognito 控制台中进行 case-by-case或测试。

当您以管理员身份创建用户时，Amazon Cognito 会为他们设置临时密码并发送欢迎或邀请消息。他们可以点击邀请消息中的链接进行首次登录、设置密码并确认其账户。以下页面说明了如何创建新用户和配置欢迎消息。有关使用用户池 API 和 Amazon SDK 或 CDK 创建用户的更多信息，请参阅AdminCreateUser。

创建用户池后，您可以使用 Amazon Web Services Management Console、以及 Amazon Command Line Interface 或 Amazon Cognito API 创建用户。您可以为用户池中的新用户创建配置文件，并通过 SMS 或电子邮件向用户发送带有注册说明的欢迎消息。

以下是管理员如何管理用户池中的用户的一些示例。

- 在 Amazon Cognito 控制台中或`AdminCreateUser`通过 API 操作创建新的用户个人资料。
- 将无密码 username-and-password、密钥和自定义身份验证流程提供给您的用户池和应用程序客户端。
- 设置用户属性值。
- 创建自定义属性。
- 在 `AdminCreateUser` API 请求中设置不可变的自定义属性的值。此功能在 Amazon Cognito 控制台中不可用。
- 指定临时密码，创建没有密码的用户，或者允许 Amazon Cognito 自动生成密码。
- 创建新用户并自动确认他们的帐户、验证他们的电子邮件地址或验证他们的电话号码。
- 通过 Amazon Web Services Management Console 或 Lambda 触发器（如自定义消息、自定义短信发送器和自定义电子邮件发件人）为新用户指定自定义短信和电子邮件邀请消息。
- 指定是否通过 SMS、电子邮件或两者发送邀请消息。
- 通过调用 `AdminCreateUser` API，并为 `RESEND` 参数指定 `MessageAction`，向现有用户重新发送欢迎消息。
- 在创建用户时@@ 禁止发送邀请消息。
- 为新用户帐户指定最多 90 天的到期时间限制。
- 允许用户自行注册或要求只能由管理员添加新用户。

管理员还可以在服务器端应用程序中使用 Amazon 凭据登录用户。有关更多信息，请参阅 API 和 SDK 身份验证的授权模型。

用户身份验证流程和创建用户

根据用户池的配置，用户管理创建的选项会有所不同。身份验证流程或可供用户登录和 MFA 的方法可以改变您创建用户的方式以及向他们发送的消息。以下是用户池中可用的一些身份验证流程。

- 用户名和密码
- 密钥

- 使用第三方登录 IdPs
- 无需密码，使用电子邮件和短信一次性密码 () OTPs
- 使用电子邮件、短信和身份验证器应用程序进行多因素身份验证 OTPs
- 使用 Lambda 触发器进行自定义身份验证

有关如何配置这些登录系数的更多信息，请参阅使用 Amazon Cognito 用户池进行身份验证。

创建没有密码的用户

如果您为用户池启用了无密码登录，则可以创建没有密码的用户。要创建没有密码的用户，必须为可用的无密码登录系数提供属性值。例如，如果您的用户池中提供电子邮件 OTP 无密码登录，则可以创建一个没有密码和电子邮件地址属性的用户。如果仅供新用户使用的身份验证流程需要密码，例如密钥或用户名-密码，则必须为每个新用户创建或生成临时密码。

创建没有密码的新用户

- 在 Amazon Cognito 控制台中选择 “不要设置密码”
- 省略或留空 `AdminCreateUser` API 请求的`TemporaryPassword`参数

系统会自动确认没有密码的用户

通常，当您创建新用户时，他们会获得临时密码并进入`FORCE_CHANGE_PASSWORD`状态。当您创建没有密码的用户时，他们会立即进入`CONFIRMED`状态。您无法向该`CONFIRMED`州的这些用户重新发送确认码。

对于没有密码的用户，邀请消息会发生变化。

默认情况下，Amazon Cognito 会向新用户发送一条邀请消息，上面写着 “`Your username is {userName} and your password is {####}.`当您创建没有密码的用户时，消息中写着 “`Your username is {userName}.`自定义您的邀请消息”，以反映您是否要为用户设置密码。在无`{####}`密码身份验证模型中省略密码变量。

当无密码因素可用时，您无法自动生成密码

如果您已将用户池配置为支持电子邮件或电话 OTP 无密码登录，则无法自动生成密码。对于每位将拥有密码的用户，在创建其个人资料时都必须设置一个临时密码。

无密码用户必须具有所有必需属性的值

在创建没有密码的用户时，只有当用户为用户池中已标记为必填的所有属性提供值时，您的请求才会成功。这适用于任何必填属性，而不仅仅是OTP交付所需的电话号码和电子邮件属性。

创建稍后将提供必填属性值的用户

您可能希望在用户池中要求属性，但是在管理性地创建用户之后、应用程序中的用户交互期间收集这些属性。管理员在使用临时密码创建用户时可以省略必填属性的值。您不能省略无密码用户的必填属性值。

缺少必填属性值和临时密码的用户在首次登录时会收到一个 NEW_PASSWORD_REQUIRED 质询。然后，他们可以为`requiredAttributes`参数中缺少的必需属性提供一个值。只有当所有必需的属性均可变时，才可以创建带有密码但没有必需属性的用户。只有当用户登录时使用的应用程序客户端可以写入所需的属性时，用户才能使用`NEW_PASSWORD_REQUIRED`挑战和必填属性值完成登录。

当您为管理员创建的用户设置永久密码时，他们的状态会更改为，`CONFIRMED`并且您的用户池不会在他们首次登录时提示他们输入新密码或必填属性。

在 Amazon Web Services Management Console中创建新用户

您可以设置用户密码要求、配置发送给用户的邀请和验证消息，以及使用 Amazon Cognito 控制台添加新用户。

设置密码策略并启用自行注册

您可以配置设置以最大限度地降低密码复杂性，以及用户是否可以在您的用户池 APIs 中使用公共方式进行注册。

配置密码策略

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “身份验证方法” 菜单并找到 “密码策略”。选择编辑。
4. 选择自定义的密码策略模式。
5. 选择密码最小长度。有关密码长度要求的限制，请参阅用户池资源配额。
6. 选择密码复杂性要求。
7. 选择管理员设置的密码应在多长时间内有效。
8. 选择保存更改。

允许自助注册

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “注册” 菜单并找到 “自助注册”。选择编辑。
4. 选择是否启用自助注册。自助注册通常用于需要在不分发客户端密钥或 Amazon Identity and Access Management (IAM) API 凭证的情况下在用户池中注册新用户的公共应用程序客户端。

   > **禁用自助注册**
   >
   > 如果您不启用自助注册，则必须通过使用 IAM API 凭证的管理 API 操作或通过联合提供商登录来创建新用户。

5. 选择保存更改。

自定义电子邮件和 SMS 消息

自定义用户消息

当您邀请用户登录、注册用户账户或登录并提示用户进行多重身份验证（MFA）时，您可以自定义 Amazon Cognito 发送给用户的消息。

> **Note**
>
> 当您在用户池中创建用户并邀请他们登录时将发送邀请消息。Amazon Cognito 将初始登录信息发送到用户的电子邮件地址或电话号码。
> 当用户在您的用户池中注册用户账户时将发送验证消息。Amazon Cognito 向用户发送代码。当用户向 Amazon Cognito 提供代码时，他们会验证自己的联系人信息并确认自己的账户以进行登录。验证代码的有效期为 24 小时。
> 当您在用户池中启用 SMS MFA，并且已配置 SMS MFA 的用户登录并提示输入 MFA 时，将发送 MFA 消息。

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “消息模板” 菜单，选择 “验证消息”、“邀请消息” 或 “MFA 消息”，然后选择 “编辑”。
4. 自定义所选消息类型的消息。

> **Note**
>
> 自定义消息时，必须包括消息模板中的所有变量。如果变量（例如，{####}）不包括在内，您的用户将没有足够的信息来完成消息操作。
> 有关更多信息，请参阅消息模板。

5. a. 验证消息
   i. 选择用于电子邮件消息的验证类型。代码验证将发送用户必须输入的数字代码。链接验证将发送一个链接，用户可以点击该链接以验证其联系人信息。用于链接消息变量中的文本显示为超链接文本。例如，使用变量 {##Click here##} 的消息模板在电子邮件中显示为单击此处。
   ii. 输入用于电子邮件消息的电子邮件主题。
   iii. 输入用于电子邮件消息的自定义电子邮件消息模板。您可以使用 HTML 自定义此模板。
   iv. 输入用于 SMS 消息的自定义 SMS 消息模板。
   v. 选择保存更改。
   b. 邀请消息
   i. 输入用于电子邮件消息的电子邮件主题。
   ii. 输入用于电子邮件消息的自定义电子邮件消息模板。您可以使用 HTML 自定义此模板。
   iii. 输入用于 SMS 消息的自定义 SMS 消息模板。
   iv. 选择保存更改。
   c. MFA 消息
   i. 输入用于 SMS 消息的自定义 SMS 消息模板。
   ii. 选择保存更改。

创建用户

创建用户

您可以从 Amazon Cognito 控制台为用户池创建新用户。通常，用户可以在设置密码后登录。要使用电子邮件地址登录，用户必须验证 `email` 属性。要使用电话号码登录，用户必须验证 `phone_number` 属性。要以管理员身份确认账户，您还可以使用 Amazon CLI 或 API，或者使用联合身份提供商创建用户个人资料。有关更多信息，请参阅 Amazon Cognito API 参考。

1. 导航到 Amazon Cognito 控制台，选择用户池。
2. 从列表中选择一个现有用户池，或创建一个用户池。
3. 选择 “用户” 菜单，然后选择 “创建用户”。
4. 检查用户池登录和安全要求以获取有关密码要求、可用的账户恢复方法和用户池的别名属性的指导。
5. 选择您希望如何发送邀请消息。选择 SMS 消息和/或电子邮件消息。要取消邀请消息，请选择 “不发送邀请”。

> **Note**
>
> 在发送邀请消息之前，请在用户池的身份验证方法菜单中 Amazon Web Services 区域 使用亚马逊简单通知服务和亚马逊简单电子邮件服务配置发件人。收件人消息和数据费率适用。Amazon SES 单独向您收取电子邮件消息费用，Amazon SNS 单独向您收取 SMS 消息费用。

6. 选择用于新用户的用户名。
7. 选择您是要为用户创建密码，还是让 Amazon Cognito 生成密码。如果用户池中提供无密码登录，则生成密码的选项不可用。任何临时密码都必须遵守用户池密码策略。
8. 选择创建。
9. 选择 “用户” 菜单，然后为用户选择用户名条目。添加和编辑用户属性和组成员资格。查看用户事件历史记录。

向用户池添加组

借助对 Amazon Cognito 用户池中组的支持，您可以创建和管理组、将用户添加到组以及从组中删除用户。使用组可创建用户集合以管理其权限或表示不同类型的用户。您可以为群组分配 Amazon Identity and Access Management (IAM) 角色来定义群组成员的权限。

您可以使用组以在用户池中创建用户集合，这通常用于为这些用户设置权限。例如，您可以为作为您网站和应用程序的读者、贡献者或编辑者的用户创建单独的组。通过使用与组关联的 IAM 角色，您还可以为那些不同的组设置不同的权限，从而只有贡献者可以将内容放置在 Amazon S3 中，并且只有编辑者可以通过 Amazon API Gateway 中的 API 发布内容。

Amazon Cognito 会为您添加到用户池中的每个 OIDC 和社交身份提供商 (IdP) 创建一个用户组。SAMI群组名称的格式为`[user pool ID]_[IdP name]`，例如`us-east-1_EXAMPLE_MYSSO`或`us-`

east-1_EXAMPLE_Google。自动生成的每个唯一的 IdP 用户配置文件都会自动添加到该群组中。关联用户不会自动添加到该群组，但您可以通过单独的流程将其个人资料添加到群组。

您可以通过、和 CLI 在用户池中 Amazon Web Services Management Console创建和管理群组。 APIs 作为开发者（使用 Amazon 证书），您可以创建、读取、更新、删除和列出用户池的群组。您还可以将用户添加到组和从组中删除用户。

在用户池中使用组不会产生额外费用。有关更多信息，请参阅 Amazon Cognito 定价。

向组分配 IAM 角色

您可以使用组通过 IAM 角色控制资源的权限。IAM 角色包含信任策略和权限策略。角色的信任策略指定谁可使用该角色。权限策略指定组成员可以访问的操作和资源。在您创建 IAM 角色时，请设置角色的信任策略以允许您的组用户担任该角色。请在角色的权限策略中，指定您希望组具有的权限。

在 Amazon Cognito 中创建组时，可以通过提供角色的 ARN 指定 IAM 角色。当组成员使用 Amazon Cognito 登录时，他们可以从身份池接收临时凭证。他们的权限由关联的 IAM 角色确定。

单个用户可处于多个组中。作为开发人员，当一个用户位于多个组中时，您可以使用以下选项自动选择 IAM 角色：

- 您可以为每个组分配优先级值。将选择优先级较高（值较低）的组，并应用其关联的 IAM 角色。
- 在通过身份池为用户请求 Amazon 凭证时，您的应用程序还可以在参数中指定角色 ARN，从可用角色中GetCredentialsForIdentityCustomRoleARN进行选择。指定的 IAM 角色必须与适用于用户的角色相匹配。

将优先级值分配到组

一个用户可属于多个组。在用户的访问令牌和 ID 令牌中，cognito:groups 声明包含用户所属的所有组的列表。cognito:roles 断言包含与这些组对应的角色列表。

由于一个用户可以属于多个组，因此可为每个组分配一个优先级。这是一个非负数值，用于指定该组相对于用户池中用户所属其它组的优先级。零是代表最高优先级的值。具有较低优先级值的组优先于具有较高或空优先级值的组。如果一个用户属于两个或更多组，则具有最低优先级值的组的 IAM 角色将应用于用户 ID 令牌中的 cognito:preferred_role 声明。

两个组可以具有相同的优先级值。如果发生这种情况，则两个组之间不存在优先情况。如果具有相同优先级值的两个组还具有相同的角色 ARN，则该角色将用于每个组中用户的 ID 令牌的 cognito:preferred_role 陈述。如果两个群组的角色不同 ARNs，则不会在用户的 ID 令牌中设置cognito:preferred_role声明。

使用组控制使用 Amazon API Gateway 的权限

您可以使用用户池中的组控制使用 Amazon API Gateway 的权限。用户所属的组包含在 `cognito:groups` 声明中的用户池的 ID 令牌和访问令牌中。您可以通过请求向 Amazon API Gateway 提交 ID 或访问令牌，并使用 Amazon Cognito 用户池授权方获取 REST API。有关更多信息，请参阅《API Gateway 开发人员指南》中的使用 Amazon Cognito 用户池作为授权方控制对 REST API 的访问。

您还可以使用自定义 JWT 授权方授权访问 Amazon API Gateway HTTP API。有关更多信息，请参阅《API Gateway 开发者指南》中的 “ APIs 使用 JWT 授权者控制 HTTP 访问权限”。

组的限制

用户组受以下限制的约束：

- 您可以创建的组的数量受 Amazon Cognito 服务配额的限制。
- 不能对组进行嵌套。
- 不能搜索组中的用户。
- 不能按名称搜索组，但可以列出组。

在 Amazon Web Services Management Console中创建新组

使用以下过程创建新组。

创建新组。

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “群组” 菜单，然后选择 “创建群组”。
5. 在 Create a group（创建组）页面的 Group name（组名称）中，为您的新组输入一个易记名称。
6. 您可以选择使用以下任意字段提供有关此组的其它信息：
   - Description（说明）– 输入有关这个新组将用于什么的详细信息。
   - Precedence（优先顺序）– Amazon Cognito 根据给定用户所属群组具有的较低优先级值，评估并应用所有群组权限。将选择优先级较低的组，并应用其关联的 IAM 角色。有关更多信息，请参阅 将优先级值分配到组。

- IAM role（IAM 角色）– 当您需要控制对资源的权限时，您可以为组分配 IAM 角色。如果您要将用户池与身份池集成，并且身份池配置为从令牌中选择角色，则 IAM role (IAM 角色) 设置将确定在用户的 ID 令牌中分配哪个角色。有关更多信息，请参阅 向组分配 IAM 角色。
- Add users to this group（将用户添加到此组）– 创建后将现有用户添加为该组的成员。

7. 选择 Create（创建）以确认。

管理和搜索用户账户

用户池可以包含数百万个用户。对于管理员来说，使用这种规模的数据集是一项挑战。Amazon Cognito 提供了用于查找和修改用户配置文件的工具。查找用户的主要方法是 Amazon Cognito 控制台的 “用户” 菜单，以及。ListUsers在检索用户信息的方法中，这些选项不会像例如那样对成本产生影响AdminGetUser。

本指南的这一部分包含有关在用户池中查找和更新用户配置文件的信息。

查看用户属性

请使用以下过程在 Amazon Cognito 控制台中查看用户属性。

查看用户属性

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单，然后在列表中选择一个用户。
5. 在用户详细信息页面，您可以在 User attributes（用户属性）中查看哪些属性与用户关联。

重置用户的密码

请使用以下过程在 Amazon Cognito 控制台重置用户的密码。

重置用户的密码

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。

4. 选择 “用户” 菜单，然后在列表中选择一个用户。
5. 在用户详细信息页面上，选择 Actions（操作）、Reset password（重置密码）。
6. 在 Reset password（重置密码）对话框中，查看信息，准备就绪后，选择 Reset（重置）。

   该操作会立即导致向用户发送确认代码，并通过将用户状态更改为 `RESET_REQUIRED` 来禁用用户的当前密码。Reset password（重置密码）代码的有效期为 1 小时。

启用、禁用和删除用户帐户

您可以删除未使用的用户配置文件，或者，如果您想暂时阻止访问，可以将其禁用。用户可以删除自己的帐户，但只有用户池管理员才能启用和禁用用户帐户。

删除的效果

用户无法使用已删除的用户帐户登录，要重新获得访问权限，必须注册或重新创建。

禁用账户的影响

当您禁用用户账户时，Amazon Cognito 会自动使所有经过身份验证的会话失效，停用用户账户进行登录，并撤消他们的访问权限和刷新令牌。当用户尝试登录您禁用的账户`User is not enabled`时，Amazon Cognito 会返回一条`invalid_request`错误消息。这种行为不会随着应用程序客户端的用户存在披露设置而改变。您可以禁用本地用户帐户和联合用户帐户的本地配置文件。当用户使用托管登录或经典托管用户界面登录，然后您禁用其帐户，然后他们尝试使用维护其身份验证会话的浏览器 Cookie 再次登录时，Amazon Cognito 会将他们重定向到登录页面。

启用账户的效果

启用账户后，用户可以立即登录账户。默认情况下，用户帐户处于启用状态。用户的属性和密码与其帐户被禁用之前相同。您的应用程序撤销的令牌，无论您是禁用用户账户还是单独撤销刷新令牌，在您启用拥有该令牌的用户账户后，都将保持无效。

Delete a user account (console)

删除用户帐户

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单，然后选择列表中用户名旁边的单选按钮。

5. 选择删除。
6. 选择 “禁用用户访问权限”。
7. 选择删除。

Delete a user account (API)

用户可以通过自助服务 access-token-authorized DeleteUserAPI 操作删除其账户。以下是`DeleteUser`请求正文的示例。

```
{
   "AccessToken": "eyJra456defEXAMPLE"
}
```

管理员可以通过 IAM 授权的 AdminDeleteUserAPI 操作删除用户帐户。以下是`AdminDeleteUser`请求正文的示例。

```
{
   "Username": "testuser",
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

Disable a user account (console)

禁用用户帐户

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单，然后在列表中选择用户的用户名。
5. 在用户详细信息页面上，选择操作，禁用用户访问权限。
6. 在由此创建的对话框中，选择 “禁用”。

Disable a user account (API)

管理员可以通过 IAM 授权的 AdminDisableUserAPI 操作禁用用户帐户。以下是`AdminDisableUser`请求正文的示例。

```
{
   "Username": "testuser",
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

Enable a user account (console)

启用用户帐户

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单，然后在列表中选择用户的用户名。
5. 在用户详细信息页面上，选择操作，启用用户访问权限。
6. 在由此创建的对话框中，选择 “启用”。

Enable a user account (API)

管理员可以通过 IAM 授权的 AdminEnableUserAPI 操作启用用户帐户。以下是`AdminEnableUser`请求正文的示例。

```
{
   "Username": "testuser",
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

搜索用户属性

如果您已创建用户池，则可以在 Amazon Web Services Management Console的 Users（用户）面板中搜索。您也可以使用 Amazon Cognito ListUsers API，它接受筛选器参数。

您可以搜索以下任何标准属性。自定义属性不可搜索。

- username（区分大小写）
- email
- phone_number

- name
- given_name
- family_name
- preferred_username
- cognito:user_status（在控制台中称为Status（状态））（区分大小写）
- status（在控制台中称为Enabled（已启用））（区分大小写）
- sub

> **Note**
>
> 您还可以使用客户端筛选条件列出用户。服务器端筛选条件匹配的属性不超过 1 个。对于高级搜索，请使用客户端筛选条件，其中包含 Amazon Command Line Interface中 `list-users` 操作的 `--query` 参数。使用客户端筛选器时，会 ListUsers 返回零个或多个用户的分页列表。您可以连续接收多个结果为零的页面。对返回的每个分页令牌重复查询，直到您收到一个空的分页令牌值，然后查看合并结果。
> 有关服务器端和客户端筛选的更多信息，请参阅《Amazon Command Line Interface 用户指南》中的筛选 Amazon CLI 输出。

使用搜索用户 Amazon Web Services Management Console

如果您已创建用户池，则可以在 Amazon Web Services Management Console的 Users（用户）面板中搜索。

Amazon Web Services Management Console 搜索始终是前缀（“以” 开头）搜索。

在 Amazon Cognito 控制台中搜索用户

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单，然后在搜索栏中输入用户名。请注意，某些属性值区分大小写 [例如，Username（用户名）]。

   您还可以通过调整搜索筛选条件来查找用户，将范围缩小到其它用户属性，如 Email（电子邮件）、Phone number（电话号码）或 Last name（姓）。

使用 ListUsers API 搜索用户

要从您的应用程序中搜索用户，请使用亚马逊 Cognito ListUsers API。此 API 使用以下参数：

- AttributesToGet：一组字符串，其中每个字符串均为将针对搜索结果中的每位用户返回的用户属性的名称。要检索所有属性，请不要包含 AttributesToGet 参数或文本字符串值为 null 的请求 AttributesToGet。
- Filter：筛选条件字符串，格式为 "AttributeName Filter-Type "AttributeValue""。筛选条件字符串中的引号必须使用反斜杠 (\) 字符进行转义。例如 "family_name = \"Reddy\""。如果筛选条件字符串为空，ListUsers 将返回用户池中的所有用户。
  - AttributeName：要搜索的属性的名称。一次只能搜索一个属性。

    > **Note**
    > 您只能搜索标准属性。自定义属性不可搜索。这是因为只有索引属性可搜索，而自定义属性不可索引。

  - Filter-Type：对于精确匹配，请使用 =，例如 given_name = "Jon"。对于前缀 (“starts with”) 匹配，请使用 ^=，例如 given_name ^= "Jon"。
  - AttributeValue：必须为每位用户匹配的属性值。
- Limit：要返回的最大用户数。
- PaginationToken：可从之前的搜索中获取更多结果的令牌。Amazon Cognito 会在一小时后让分页令牌过期。
- UserPoolId：应对其执行搜索的用户池的用户池 ID。

所有搜索都区分大小写。搜索结果按以 AttributeName 字符串命名的属性进行升序排列。

使用 ListUsers API 的示例

以下示例将返回所有用户并包括所有属性。

```
{
    "AttributesToGet": null,
    "Filter": "",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
```

```
}
```

以下示例将返回电话号码以“+1312”开头的所有用户并包括所有属性。

```
{
    "AttributesToGet": null,
    "Filter": "phone_number ^= \"+1312\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

以下示例将返回姓氏为“Reddy”的前 10 位用户。对于每个用户，搜索结果包含用户的名字、电话号码和电子邮件地址。如果用户池中有 10 个以上相匹配的用户，则响应将包含一个分页标记。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "UserPoolId": "us-east-1_samplepool"
}
```

如果上一示例返回分页标记，则以下示例将返回与同一筛选条件字符串相匹配的接下来的 10 位用户。

```
{
    "AttributesToGet": [
        "given_name",
        "phone_number",
        "email"
    ],
    "Filter": "family_name = \"Reddy\"",
    "Limit": 10,
    "PaginationToken": "pagination_token_from_previous_search",
    "UserPoolId": "us-east-1_samplepool"
}
```

密码、账户恢复和密码策略

所有登录到用户池的用户（甚至是联合用户）都为其用户配置文件分配了密码。本地用户和关联用户在登录时必须提供密码。联合用户不使用用户池密码，而是使用其身份提供者（IdP）登录。您可以允许用户自行重置密码、以管理员身份重置或更改密码，以及设置密码复杂度和历史策略。

Amazon Cognito 不以明文形式存储用户密码。而是通过用户特定的加密盐来存储每个用户密码的哈希值。因此，您无法从用户池中的用户配置文件检索现有密码。作为一项最佳实践，请不要在任何地方存储明文用户密码。当用户忘记密码时，执行密码重置。

密码重置和恢复

用户忘记了自己的密码。您可能希望他们能够自己重置密码，或者您可能希望由管理员为他们重置密码。Amazon Cognito 用户池有这两种模式的选项。指南的这一部分介绍用户池设置和用于密码重置的 API 操作。

ForgotPasswordAPI 操作和托管登录选项忘记密码了吗？ 向用户发送验证码，当他们确认自己拥有正确的密码时，他们便有机会设置新密码ConfirmForgotPassword。这是自助式密码恢复模式。

恢复未经验证的用户

您可以向已验证其电子邮件地址或电话号码的用户发送恢复消息。如果他们没有已确认的辅助电子邮件或电话，则用户池管理员可以将他们的电子邮件地址或电话号码标记为已验证。在 Amazon Cognito 控制台中编辑用户的用户属性，然后选中 “将电话号码标记为已验证” 或 “将电子邮件地址标记为已验证” 旁边的复选框。您也可以在AdminUpdateUserAttributes请求中将`email_verified`或设置`phone_number_verified`为 true。对于新用户，ResendConfirmationCodeAPI 操作会向他们的电子邮件地址或电话号码发送新的验证码，他们就可以完成自助确认和验证。

以管理员身份重置密码

AdminSetUserPassword和 AdminResetUserPasswordAPI 操作是管理员启动的密码重置方法。`AdminSetUserPassword`设置临时或永久密码，并以与相同的方式`AdminResetUserPassword`向用户发送密码重置代码。`ForgotPassword`

配置密码重置和恢复

Amazon Cognito 会自动从您在控制台中创建用户池时选择的必需属性和登录选项中选择您的账户恢复选项。您可以修改这些默认设置。

用户首选 MFA 方法会影响他们可用于恢复密码的方法。首选 MFA 方式为电子邮件的用户无法通过电子邮件接收密码重置代码。首选 MFA 方式为短信的用户无法通过短信接收密码重置代码。

当用户不符合条件，无法使用首选密码重置方法时，您的密码恢复设置必须提供替代选项。例如，您的恢复机制可能将电子邮件列为第一优先选项，而电子邮件 MFA 可能是您的用户池中的一个选项。在这种情况下，添加短信消息账户恢复作为第二个选项，或者使用管理 API 操作为这些用户重置密码。

> **Note**
>
> 用户无法通过相同的电子邮件地址或电话号码接收 MFA 和密码重置码。如果他们使用电子邮件中的一次性密码 (OTPs) 进行 MFA，则必须使用 SMS 消息进行账户恢复。如果他们使用来 OTPs 自 SMS 消息的 MFA，则必须使用电子邮件进行账户恢复。在具有 MFA 的用户池中，如果用户有电子邮件地址的属性但没有电话号码，或者有电话号码但没有电子邮件地址，则他们可能无法完成自助密码恢复。
> 要防止出现用户无法在使用此配置的用户池中重置密码的状态，请根据需要设置email和phone_number属性。或者，您可以设置在用户注册或管理员创建用户配置文件时始终收集和设置这些属性的流程。当用户同时拥有这两个属性时，Amazon Cognito 会自动向目标发送密码重置代码，而该代码不是用户的 MFA 因子。

以下过程在用户池中配置自助服务帐户恢复。

Configure self-service password reset (API/SDK)

该AccountRecoverySetting参数是用户池参数，用于设置用户在 ForgotPasswordAPI 请求中或选择 “忘记密码？” 时可以用来恢复密码的方法 在托管登录中。 ForgotPassword向经过验证的电子邮件或经过验证的电话号码发送恢复码。恢复代码的有效期为 1 小时。当您为用户池指定 AccountRecoverySetting 时，Amazon Cognito 会根据您设置的优先级选择代码发送目标。

当您定义 AccountRecoverySetting 并且用户配置了 SMS MFA 时，不能将 SMS 用作账户恢复机制。此设置的优先级由最高优先级确定。1Amazon Cognito 仅向其中一种指定方法发送验证。以下示例AccountRecoverySetting将电子邮件地址设置为账户恢复代码的主要目的地，如果用户没有电子邮件地址属性，则回退到 SMS 消息。

```
"AccountRecoverySetting": {
   "RecoveryMechanisms": [
      {
         "Name": "verified_email",
         "Priority": 1
      },
      {
         "Name": "verified_phone_number",
```

```
            "Priority": 2
        }
    ]
}
```

该值admin_only会关闭自助服务账户恢复，而是要求用户联系管理员进行密码重置。您不能将admin_only 与任何其他账户恢复机制一起使用。以下 e

```
"AccountRecoverySetting": {
   "RecoveryMechanisms": [
      {
         "Name": "admin_only",
         "Priority": 1
      }
   ]
}
```

如果您未指定AccountRecoverySetting，Amazon Cognito 会先将恢复码发送到经过验证的电话号码，如果用户没有电话号码属性，则会将恢复码发送到经过验证的电子邮件地址。

有关 AccountRecoverySetting 的更多信息，请参阅 UpdateUserPool和 CreateUserPool。

Configure self-service password reset (console)

从用户池的登录菜单中配置账户恢复和密码重置选项。

设置用户帐户恢复

1. 登录 Amazon Cognito 控制台。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “登录” 菜单。找到 “用户帐户恢复”，然后选择 “编辑”
5. 要允许用户重置自己的密码，请选择启用自助服务帐户恢复。
6. 为用户池发送给用户的密码恢复代码配置传送方式。在 “用户帐户恢复消息的传送方式” 下，选择一个可用选项。作为最佳实践，请选择具有辅助消息发送方法的选项，例如电子邮件（如果可用），否则选择短信。借助二级交付方式，Amazon Cognito 可以要求用户使用与 MFA 不同的媒介来重置密码。
7. 选择保存更改。

忘记密码行为

在给定时间内，作为忘记密码和操作的一部分，我们允许用户尝试请求或输入密码重置码 5 到 20 次。confirm-forgot-password确切的值取决于与请求关联的风险参数。请注意，这种行为可能会发生变化。

添加用户池密码要求

作为用户池的最佳安全实践，应该设置强大、复杂的密码。特别是在对互联网开放的应用程序中，弱密码会将用户的凭证暴露给会猜测密码并尝试访问您的数据的系统。密码越复杂，就越难猜出。Amazon Cognito 为注重安全的管理员提供了其他工具 ACLs，例如威胁防护和Amazon WAF 网络，但是您的密码策略是用户目录安全的核心要素。

Amazon Cognito 用户池中本地用户的密码不会自动过期。妥善的做法是在外部系统中记录用户密码重置的时间、日期和元数据。通过记录密码使用期限的外部日志，您的应用程序或 Lambda 触发器可以查找用户的密码使用期限，并在给定时间后要求重置。

您可以将用户池配置为要求密码具有最低复杂性，以符合您的安全标准。复杂密码的最小长度为至少八个字符。还必须包括大写字母、数字和特殊字符的组合。

使用 Essentials 或 Plus 功能层，您还可以设置密码重复使用策略。您可以阻止用户将其新密码重置为与其当前密码相同，也不得与最多 23 个以前的其他密码中的任何一个相同，即用户不能将新密码设置为这 24 个密码中的任何一个。

设置用户池密码策略

1. 创建用户池并导航到 “配置安全要求” 步骤，或者访问现有用户池并导航到 “身份验证方法” 菜单。
2. 导航到密码策略。
3. 选择密码策略模式。Cognito 默认使用推荐的最低设置来配置您的用户池。您也可以选择一项自定义密码策略。
4. 设置密码最小长度。所有用户都必须使用长度大于或等于这个值的密码进行注册或创建。您可以将这个最小值设置为 99，但用户可以设置最长 256 个字符的密码。
5. 在密码要求下配置密码的复杂性规则。选择您希望在每个用户的密码中至少包含一个的字符类型（数字、特殊字符、大写字母和小写字母）。

   可以要求密码中至少包含以下字符之一：在 Amazon Cognito 确认密码中包含所需的最少字符后，用户的密码可以包含任何类型的额外字符，但不得超过最大密码长度。

   - 大写和小写基本拉丁字母

- 数字
- 以下特殊字符。

```
^ $ * . [ ] { } ( ) ? " ! @ # % & / \ , > < ' : ; | _ ~ ` = + -
```

- 非前导、非结尾的空格字符。

6. 为管理员设置的临时密码到期时间设置一个值。超过此期限，您通过 Amazon Cognito 控制台或 `AdminCreateUser` 创建的新用户将无法登录和设置新密码。使用临时密码登录后，他们的用户账户永远不会过期。要在 Amazon Cognito 用户池 API 中更新密码时长，请在您的CreateUserPool或 UpdateUserPoolAPI 请求TemporaryPasswordValidityDays 中为设置一个值。
7. 为防止使用之前的密码设置一个值（如果有）。要使用此功能，请在用户池中选择 “必备” 或 “增强” 功能层。此参数的值是在用户重置密码时阻止新密码匹配的先前密码数。

要重置已过期用户账户的访问权限，请执行以下操作之一：

- 删除用户配置文件并创建新的用户配置文件。
- 在 AdminSetUserPasswordAPI 请求中设置新的永久密码。
- 在 AdminResetUserPasswordAPI 请求中生成新的确认码。

将用户导入一个用户池

您可以使用以下两种方式将用户从现有用户目录或用户数据库导入或迁移到 Amazon Cognito 用户池中。您可以利用用户迁移 Lambda 触发器，在用户首次使用 Amazon Cognito 登录时迁移用户。借助这种方法，用户可以继续使用其现有的密码，不必在迁移到用户池后重置密码。或者，您可以上传 CSV 文件 (包含所有用户的用户配置文件属性)，批量迁移用户。以下各部分分别介绍了这两种方法。

更多资源

- 将用户迁移到 Amazon Cognito 用户池的方法
- Amazon re: inforce 2023-迁移到亚马逊 Cognito

主题

- 利用用户迁移 Lambda 触发器导入用户
- 通过 CSV 文件将用户导入用户池中

利用用户迁移 Lambda 触发器导入用户

使用这种方法，当用户首次登录您的应用程序或请求重置密码时，您可以将用户从现有用户目录无缝迁移到用户池。向您的用户池添加一个 迁移用户 Lambda 触发器 函数，它会接收有关尝试登录的用户的元数据，并从外部身份源返回用户配置文件信息。有关此 Lambda 触发器的详细信息以及示例代码（包括请求和响应参数），请参阅迁移用户 Lambda 触发器参数。

在开始迁移用户之前，请在您的 Amazon Web Services 账户中创建一个用户迁移 Lambda 函数，并将该 Lambda 函数设置为您的用户池中的用户迁移触发器。向您的 Lambda 函数添加授权策略，该策略仅允许 Amazon Cognito 服务账户主体 `cognito-idp.amazonaws.com` 调用该 Lambda 函数，并且只能在您自己的用户池的上下文中进行。有关更多信息，请参阅对 Amazon Lambda 使用基于资源的策略（Lambda 函数策略）。

登录流程

1. 用户打开您的应用程序，然后使用 Amazon Cognito 用户池 API 或通过托管登录进行登录。有关如何简化使用 Amazon Cogn APIs ito 登录的更多信息，请参阅。将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成
2. 您的应用程序将用户名和密码发送至 Amazon Cognito。如果您的应用程序具有使用 Amazon SDK 构建的自定义登录界面，则您的应用程序必须使用InitiateAuth或AdminInitiateAuth与`USER_PASSWORD_AUTH`或`ADMIN_USER_PASSWORD_AUTH`流程。当您的应用使用其中一个流程时，开发工具包会将密码发送到服务器。

   > **Note**
   >
   > 在添加用户迁移触发器之前，请在您的应用程序客户端的设置中激活 `USER_PASSWORD_AUTH` 或 `ADMIN_USER_PASSWORD_AUTH` 流程。您必须使用这些流程而不是默认 `USER_SRP_AUTH` 流程。Amazon Cognito 必须向您的 Lambda 函数发送密码，以便它可以验证您的用户在另一个目录中的身份验证。SRP 会在您的 Lambda 函数中隐藏您用户的密码。

3. Amazon Cognito 检查提交的用户名是否与用户池中的用户名或别名匹配。您可以将用户的电子邮件地址、电话号码或首选用户名设置为用户池中的别名。如果用户不存在，Amazon Cognito 会将参数（包括用户名和密码）发送到您的 迁移用户 Lambda 触发器 函数。
4. 您的 迁移用户 Lambda 触发器 函数使用您的现有用户目录或用户数据库检查用户，或验证用户身份。该函数返回 Amazon Cognito 存储在用户池的用户配置文件中的用户属性。仅当提交的用户名与别名属性匹配时，您才能返回 `username` 参数。如果您希望用户继续使用其现有密码，您的函数

将在 Lambda 响应中将属性 `finalUserStatus` 设置为 `CONFIRMED`。您的应用程序必须返回 迁移用户 Lambda 触发器参数 中显示的所有 `"response"` 参数。

> **Important**
>
> 不要在您的用户迁移 Lambda 代码中记录整个请求事件对象。此请求事件对象包括用户的密码。如果您不对日志进行消毒，则密码会显示在 CloudWatch 日志中。

5. Amazon Cognito 在您的用户池中创建用户配置文件，并将令牌返回您的应用程序客户端。
6. 您的应用程序执行令牌接收，接受用户身份验证，然后继续处理请求的内容。

迁移用户后，请使用 `USER_SRP_AUTH` 进行登录。安全远程密码 (SRP) 协议不会通过网络发送密码，并为您在迁移期间使用的 `USER_PASSWORD_AUTH` 流程提供安全优势。

如果迁移期间出现错误（包括客户端设备或网络问题），您的应用程序会从 Amazon Cognito 用户池 API 接收错误响应。发生这种情况时，Amazon Cognito 可能会也可能不会在您的用户池中创建用户账户。然后，用户应尝试再次登录。如果登录反复失败，请尝试在您的应用程序中使用忘记密码流程重置用户密码。

忘记密码流程还会使用 `UserMigration_ForgotPassword` 事件源调用您的 迁移用户 Lambda 触发器 函数。由于用户在请求密码重置时没有提交密码，因此 Amazon Cognito 在发送到您的 Lambda 函数的事件中不包含密码。您的函数只能在现有用户目录中查找用户并返回属性，以添加到用户池中的用户配置文件中。在您的函数完成其调用并将其响应返回给 Amazon Cognito 后，您的用户群体将通过电子邮件或 SMS 发送密码重置代码。在您的应用程序中，提示您的用户输入确认码和新密码，然后ConfirmForgotPassword通过 API 请求将该信息发送给 Amazon Cognito。您还可以在托管登录中使用内置页面来处理忘记密码流程。

其他资源

- 将用户迁移到 Amazon Cognito 用户池的方法

通过 CSV 文件将用户导入用户池中

如果您有外部身份存储，并且有时间为新的本地用户准备用户池，那么在迁移到 Amazon Cognito 用户池，选择从逗号分隔值（CSV）文件批量导入用户既省时省力，又可降低成本。CSV 文件导入是先下载和填入模板文件，然后在导入任务中将该文件移交给用户池的过程。您可以使用 CSV 导入来快速创建测试用户。您还可以通过编程的方式，使用读取 API 请求从外部身份存储中获取数据，然后解析这些数据的详细信息和属性，再将它们写入到文件中。

导入过程会设置所有用户属性的值，不过 password 除外。不支持导入密码，因为安全妥善做法要求密码不能为纯文本，而我们不支持导入哈希。这意味着，用户必须在首次登录时更改密码。使用此方法导入用户时，用户处于 RESET_REQUIRED 状态。

从 CSV 导入用户最省力的方法是在用户池中激活无密码登录。借助电子邮件地址和电话号码属性以及正确的用户池配置，用户可以在导入任务完成后立即使用电子邮件或短信一次性密码 (OTPs) 登录。有关更多信息，请参阅 要求导入的用户重置密码。

您也可以使用以下方式设置用户的密码 AdminSetUserPassword将Permanent参数设置为的 API 请求true。CSV 导入不会计入用户池中按月计费的活跃用户 (MAUs)。但是，确实会生成密码重置操作。 MAUs要在导入大量使用密码但可能不会立即处于活动状态的用户时管理成本，请将您的应用程序设置为在用户登录并接受RESET_REQUIRED质询时提示他们输入新密码。

> **Note**
>
> 每个用户的创建日期就是将该用户导入用户池中的日期。创建日期不是导入的属性之一。

创建用户导入任务的步骤

1. 在 Amazon Identity and Access Management (IAM) 控制台中创建 Amazon L CloudWatch ogs 角色。
2. 创建用户导入 .csv 文件。
3. 创建并运行用户导入任务。
4. 上传用户导入 .csv 文件。
5. 启动并运行用户导入任务。
6. CloudWatch 用于查看事件日志。
7. 要求导入的用户重置密码。

更多资源

- Cognito 用户配置文件导出参考架构，用于在用户池之间导出用户账户

主题

- 创建日 CloudWatch 志 IAM 角色
- 创建用户导入 CSV 文件

- 创建并运行 Amazon Cognito 用户池导入任务
- 在 CloudWatch 控制台中查看用户池导入结果
- 要求导入的用户重置密码

创建日 CloudWatch 志 IAM 角色

如果您使用的是 Amazon Cognito CLI 或 API，则需要创建一个 CloudWatch IAM 角色。以下过程介绍如何创建一个 IAM 角色，Amazon Cognito 可以使用该角色将导入任务的结果写入日志。 CloudWatch

> **Note**
>
> 在 Amazon Cognito 控制台中创建导入作业时，您可以同时创建 IAM 角色。当您选择 Create a new IAM role（创建新 IAM 角色）时，Amazon Cognito 会自动对该角色应用相应的信任策略和 IAM policy。

创建用于用户池导入的 CloudWatch Logs IAM 角色 (Amazon CLI，API)

1. 登录 Amazon Web Services Management Console 并打开 IAM 控制台，网址为https://console.aws.amazon.com/iam/。
2. 为创建新的 IAM 角色 Amazon Web Services 服务。有关详细说明，请参阅《Amazon Identity and Access Management 用户指南》中的为 Amazon Web Services 服务创建一个角色。
   a. 当您为 Trusted entity type（可信实体类型）选择 Use case（使用案例）时，请选择任意服务。Amazon Cognito 目前未在服务使用案例中列出。
   b. 在 Add permissions（添加权限）屏幕中，选择 Create policy（创建策略）并插入以下策略声明。例如，*REGION* Amazon Web Services 区域 替换为用户池中的`us-east-1`。例如*ACCOUNT*，用您的 Amazon Web Services 账户 身份证替换`111122223333`。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogGroup",
                "logs:CreateLogStream",
                "logs:DescribeLogStreams",
```

```
                "logs:PutLogEvents"
            ],
            "Resource": [
                "arn:aws:logs:REGION:ACCOUNT:log-group:/aws/cognito/*"
            ]
        }
    ]
}
```

3. 由于您在创建角色时没有选择 Amazon Cognito 作为可信实体，因此您现在必须手动编辑该角色的信任关系。在 IAM 控制台的导航窗格中选择 Roles（角色），然后选择您创建的新角色。
4. 选择 Trust relationships（信任关系）选项卡。
5. 选择编辑信任策略。
6. 将以下策略声明粘贴到 Edit trust policy（编辑信任策略）中，替换任何现有文本：

```
{
        "Version": "2012-10-17",
        "Statement": [
            {
                "Effect": "Allow",
                "Principal": {
                    "Service": "cognito-idp.amazonaws.com"
                },
                "Action": "sts:AssumeRole"
            }
        ]
    }
```

7. 选择更新策略。
8. 记下 角色 ARN。您在创建导入作业时需要此 ARN。

创建用户导入 CSV 文件

您必须先创建逗号分隔值（CSV，Comma-Separated Value）文件，在其中包含要导入的用户及其属性，然后才能将现有用户导入用户群体中。从用户群体中，您可以检索其标头反映了您的用户群体的属性架构的用户导入文件。然后，您可以插入符合 设置 CSV 文件的格式 中的格式要求的用户信息。

下载 CSV 文件标头（控制台）

使用以下步骤下载 CSV 标头文件。

下载 CSV 文件标头

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单。
5. 在 Import users（导入用户）部分中，选择 Create an import job（创建导入作业）。
6. 在 Upload CSV（上传 CSV）下，选择 template.csv 链接并下载 CSV 文件。

下载 CSV 文件标头（Amazon CLI）

要获取正确标头的列表，请运行以下 CLI 命令，其中*USER_POOL_ID*是要将用户导入到的用户池的用户池标识符：

```
aws cognito-idp get-csv-header --user-pool-id "USER_POOL_ID"
```

示例响应:

```
{
    "CSVHeader": [
        "name",
        "given_name",
        "family_name",
        "middle_name",
        "nickname",
        "preferred_username",
        "profile",
        "picture",
        "website",
        "email",
        "email_verified",
        "gender",
        "birthdate",
        "zoneinfo",
        "locale",
        "phone_number",
        "phone_number_verified",
        "address",
        "updated_at",
        "cognito:mfa_enabled",
```

```
        "cognito:username"
    ],
    "UserPoolId": "USER_POOL_ID"
}
```

设置 CSV 文件的格式

下载的用户导入 CSV 标头文件类似于以下字符串。它还包括您已添加到用户群体的所有自定义属性。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
```

编辑 CSV 文件，以使文件包含此标头和用户的属性值，并根据以下规则设置格式：

> **Note**
>
> 有关属性值（如电话号码的正确格式）的更多信息，请参阅使用用户属性。

- 文件的第一行是已下载的包含用户属性名称的标头行。
- CSV 文件中列的顺序不重要。
- 第一行之后的每一行都包含用户的属性值。
- 标头中的所有列都必须存在，但您不需要在每一列中提供值。
- 以下属性为必需属性：
  - cognito:username
  - cognito:mfa_enabled
  - email_verified 或 phone_number_verified
    - 每个用户至少有一个自动验证属性必须为 `true`。自动验证的属性是新用户加入您的用户群体时，Amazon Cognito 自动向其发送验证码的电子邮件地址或电话号码。
    - 用户池必须至少有一个自动验证属性，要么是 email_verified，要么是 phone_number_verified。如果用户池没有自动验证属性，则导入任务不会启动。
    - 如果用户池只有一个自动验证属性，则该属性必须针对每个用户进行验证。例如，如果用户池只有 phone_number 为自动验证属性，则每个用户的 phone_number_verified 值都必须为 `true`。

> **Note**
>
> 对于重置其密码的用户，用户必须拥有经过验证的电子邮件或电话号码。Amazon Cognito 将包含重置密码代码的消息发送到 CSV 文件中指定的电子邮件或电话号码。如果将消息

发送到电话号码，则通过 SMS 消息发送。有关更多信息，请参阅 在注册时验证联系人信息。

- email（如果 email_verified 为 `true`）
- phone_number（如果 phone_number_verified 为 `true`）
- 创建用户池时标记为必需的所有属性

- 字符串式的属性值不 应该用引号括起来。
- 如果属性值包含逗号，则您必须在逗号前使用反斜杠 (\)。这是因为 CSV 文件中的字段使用逗号分隔。
- CSV 文件内容应采用不含字节顺序标记的 UTF-8 格式。
- cognito:username 字段是必填项，并且在用户池中必须是唯一的。它可以是任何 Unicode 字符串。但是，它不能包含空格或制表符。
- 出生日期值（如果存在）必须采用以下格式*mm/dd/yyyy*。也就是说，如果生日日期为 1985 年 2 月 1 日，则必须编码为 **02/01/1985**。
- cognito:mfa_enabled 字段为必填字段。如果您已将用户池设置为需要进行多重验证 (MFA)，则所有用户的此字段都必须为 `true`。如果您已将 MFA 设置为关闭，则所有用户的此字段都必须为 `false`。如果您已将 MFA 设置为可选，则此字段可以是 `true` 或 `false`，但不能为空。
- 最大长度为 16000 个字符。
- CSV 文件的最大大小为 100MB。
- 文件中的最大行（用户）数为 500000。此最大值不包括标题行。
- updated_at 字段值应为纪元时间（用秒表示），例如：**1471453471**。
- 属性值中的所有前导空格或尾部空格均应去除。

以下列表是没有自定义属性的用户群体的 CSV 导入文件示例。您的用户群体架构可能与此示例有所不同。在这种情况下，您必须在从用户群体下载的 CSV 模板中提供测试值。

```
cognito:username,name,given_name,family_name,middle_name,nickname,preferred_username,profile,pi
John,,John,Doe,,,,,,,johndoe@example.com,TRUE,,02/01/1985,,,+12345550100,TRUE,123 Any
 Street,,FALSE
Jane,,Jane,Roe,,,,,,,janeroe@example.com,TRUE,,01/01/1985,,,+12345550199,TRUE,100 Main
 Street,,FALSE
```

创建并运行 Amazon Cognito 用户池导入任务

本节介绍如何使用 Amazon Cognito 控制台和 Amazon Command Line Interface ()Amazon CLI创建和运行用户池导入任务。

主题

- 从 CSV 文件导入用户（控制台）
- 导入用户（Amazon CLI）

从 CSV 文件导入用户（控制台）

以下过程介绍了如何从 CSV 文件导入用户。

从 CSV 文件导入用户（控制台）

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “用户” 菜单。
5. 在 Import users（导入用户）部分中，选择 Create an import job（创建导入作业）。
6. 在 Create import job（创建导入作业）页面上，输入 Job name（作业名称）。
7. 选择 Create a new IAM role（创建新的 IAM 角色）或者 Use an existing IAM role（使用现有 IAM 角色）。
   a. 如果您选择 Create a new IAM role（创建新的 IAM 角色），请输入新角色的名称。Amazon Cognito 将自动创建具有正确权限和信任关系的角色。创建导入作业的 IAM 主体必须具有创建 IAM 角色的权限。
   b. 如果您选择 Use an existing IAM role（使用现有 IAM 角色），请从 IAM role selection（IAM 角色选择）下的列表中选择一个角色。此角色必须具有 创建日 CloudWatch 志 IAM 角色 中所述的权限和信任策略。
8. 在上传 CSV 下，选择选择文件并附加您准备的 CSV 文件。
9. 选择 Create job（创建作业）可提交作业，但稍后再启动。选择 Create and start job（创建并启动作业）可提交您的作业并立即启动。
10. 如果您创建了作业但未启动作业，则可以稍后再启动。在 “导入用户” 下的 “用户” 菜单中，选择您的导入任务，然后选择 “开始”。您也可以从 Amazon SDK 提交 StartUserImportJobAPI 请求。

11. 在 “导入用户” 下的 “用户” 菜单中监控用户导入任务的进度。如果您的作业不成功，则可以选择 Status（状态）值。要了解更多详细信息，请选择查看 CloudWatch 日志以了解更多详细信息，然后在 CloudWatch 日志控制台中查看所有问题。

导入用户（Amazon CLI）

以下 CLI 命令可用于将用户导入到用户池：

- `create-user-import-job`
- `get-csv-header`
- `describe-user-import-job`
- `list-user-import-jobs`
- `start-user-import-job`
- `stop-user-import-job`

要获取这些命令的命令行选项列表，请使用 `help` 命令行选项。例如：

```
aws cognito-idp get-csv-header help
```

创建用户导入任务

创建 CSV 文件后，通过运行以下 CLI 命令创建用户导入任务，其中*JOB_NAME*是您为任务选择的名称，*USER_POOL_ID*是要向其中添加新用户的用户池的用户池 ID，*ROLE_ARN*也是您在中收到的角色 ARN：创建日 CloudWatch 志 IAM 角色

```
aws cognito-idp create-user-import-job --job-name "JOB_NAME" --user-pool-id
 "USER_POOL_ID" --cloud-watch-logs-role-arn "ROLE_ARN"
```

响应中*PRE_SIGNED_URL*返回的有效期为 15 分钟。在此之后，它将过期，而您必须创建新的用户导入任务以获取新的 URL。

Example 回应：

```
{
    "UserImportJob": {
        "Status": "Created",
```

```
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

用户导入任务的状态值

在对用户导入命令的响应中，您将看到以下 `Status` 值当中的其中一个值：

- `Created` – 任务已创建但未启动。
- `Pending` – 转换状态。您已启动任务，但它尚未开始导入用户。
- `InProgress` – 任务已启动，正在导入用户。
- `Stopping` – 您已停止任务，但任务尚未停止导入用户。
- `Stopped` – 您已停止任务，且任务已停止导入用户。
- `Succeeded` – 任务已成功完成。
- `Failed` – 任务因错误而停止。
- `Expired` – 您创建了一个任务，但未在 24-48 小时内启动任务。与任务关联的所有数据已删除，且任务无法启动。

上传 CSV 文件

使用以下 `curl` 命令将包含用户数据的 CSV 文件上传到您从 `create-user-import-job` 命令的响应中获取的预签名 URL。

```
curl -v -T "PATH_TO_CSV_FILE" -H "x-amz-server-side-encryption:aws:kms"
 "PRE_SIGNED_URL"
```

在此命令的输出中，查找 `"We are completely uploaded and fine"` 这一短语。此短语表示文件已成功上传。运行导入任务后，您的用户池不会将信息保留在导入文件中。它们完成或过期后，Amazon Cognito 会删除您上传的 CSV 文件。

描述用户导入任务

要获取用户导入任务的描述，请使用以下命令，其中*USER_POOL_ID*是您的用户池 ID，*JOB_ID*是您创建用户导入任务时返回的任务 ID。

```
aws cognito-idp describe-user-import-job --user-pool-id "USER_POOL_ID" --job-id
 "JOB_ID"
```

Example 示例响应:

```
{
    "UserImportJob": {
        "Status": "Created",
        "SkippedUsers": 0,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl": "PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn":"ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

在前面的示例输出中，*PRE_SIGNED_URL*是您将 CSV 文件上传到的网址。*ROLE_ARN*是您在创建角色时收到的 CloudWatch 日志角色 ARN。

列出用户导入任务

要列出用户导入任务，请使用以下命令：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 2
```

Example 示例响应:

```
{
    "UserImportJobs": [
        {
            "Status": "Created",
            "SkippedUsers": 0,
```

```
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "JobName": "JOB_NAME",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 0,
            "CreationDate": 1470957431.965
        },
        {
            "CompletionDate": 1470954227.701,
            "StartDate": 1470954226.086,
            "Status": "Failed",
            "UserPoolId": "USER_POOL_ID",
            "ImportedUsers": 0,
            "SkippedUsers": 0,
            "JobName": "JOB_NAME",
            "CompletionMessage": "Too many users have failed or been skipped during the
 import.",
            "JobId": "JOB_ID",
            "PreSignedUrl":"PRE_SIGNED_URL",
            "CloudWatchLogsRoleArn":"ROLE_ARN",
            "FailedUsers": 5,
            "CreationDate": 1470953929.313
        }
    ],
    "PaginationToken": "PAGINATION_TOKEN"
}
```

任务按创建日期 (从近到远) 排列。第二个任务之后的*PAGINATION_TOKEN*字符串表示此列表命令还有其他结果。要列出更多结果，请使用 `--pagination-token` 选项，如下所示：

```
aws cognito-idp list-user-import-jobs --user-pool-id "USER_POOL_ID" --max-results 10 --
pagination-token "PAGINATION_TOKEN"
```

启动用户导入任务

要启动用户导入任务，请使用以下命令：

```
aws cognito-idp start-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

每个账户每次只能有一个导入任务处于活动状态。

Example 示例响应:

```
{
    "UserImportJob": {
        "Status": "Pending",
        "StartDate": 1470957851.483,
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957431.965
    }
}
```

停止用户导入任务

要停止正在进行的用户导入任务，请使用以下命令。停止任务后，无法重新启动该任务。

```
aws cognito-idp stop-user-import-job --user-pool-id "USER_POOL_ID" --job-id "JOB_ID"
```

Example 示例响应:

```
{
    "UserImportJob": {
        "CompletionDate": 1470958050.571,
        "StartDate": 1470958047.797,
        "Status": "Stopped",
        "UserPoolId": "USER_POOL_ID",
        "ImportedUsers": 0,
        "SkippedUsers": 0,
        "JobName": "JOB_NAME",
        "CompletionMessage": "The Import Job was stopped by the developer.",
        "JobId": "JOB_ID",
        "PreSignedUrl":"PRE_SIGNED_URL",
        "CloudWatchLogsRoleArn": "ROLE_ARN",
        "FailedUsers": 0,
        "CreationDate": 1470957972.387
    }
```

```
}
```

在 CloudWatch 控制台中查看用户池导入结果

您可以在 Amazon CloudWatch 控制台中查看导入任务的结果。

主题

- 查看结果
- 解析结果

查看结果

以下步骤介绍了如何查看用户池导入结果。

查看用户池导入结果的步骤

1. 登录 Amazon Web Services Management Console 并打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 选择 Logs (日志)。
3. 为用户池导入任务选择日志组。日志组名称的形式为 `/aws/cognito/userpools/`*`USER_POOL_ID`*`/`*`USER_POOL_NAME`*。
4. 为刚运行的用户导入任务选择日志。日志名称的格式为*`JOB_ID`*/*`JOB_NAME`*。日志中的结果按行号引用用户。日志中不会写入用户数据。对于每个用户，都将出现类似于以下内容的行：
   - `[SUCCEEDED] Line Number 5956 - The import succeeded.`
   - `[SKIPPED] Line Number 5956 - The user already exists.`
   - `[FAILED] Line Number 5956 - The User Record does not set any of the auto verified attributes to true. (Example: email_verified to true).`

解析结果

成功导入的用户的状态设置为 “PasswordReset”。

在以下情况下，将不会导入用户，但导入任务将继续：

- 自动验证属性未设置为 `true`。
- 用户数据与架构不匹配。

- 由于内部错误，无法导入用户。

在以下情况下，导入任务将失败：

- 无法担任 CloudWatch Amazon Logs 角色，该角色的访问策略不正确，或者已被删除。
- 用户池已删除。
- Amazon Cognito 无法解析 .csv 文件。

要求导入的用户重置密码

如果您的用户池仅提供基于密码的登录，则用户必须在导入密码后重置密码。他们第一次登录时可以输入任何密码。Amazon Cognito 会提示他们在 API 响应您的应用程序的登录请求时输入新密码。

如果您的用户池具有无密码身份验证因子，Amazon Cognito 会默认为导入用户的身份验证系数。他们不会被提示输入新密码，他们可以立即使用无密码电子邮件或 SMS OTP 登录。您还可以提示用户设置密码，以便他们可以完成其他登录方法，例如用户名密码和密钥。以下条件适用于用户导入后的无密码登录。

1. 您必须导入具有与可用无密码登录系数相对应的属性的用户。如果用户可以使用电子邮件地址登录，则必须导入一个`email`属性。如果是电话号码，则必须导入一个`phone_number`属性。如果两者兼而有之，则为任一属性导入一个值。
2. 通常，用户在必须重置密码的`RESET_REQUIRED`状态下进行导入。如果导入时能够使用无密码因素登录，则 Amazon Cognito 会将其状态设置为。`CONFIRMED`

有关无密码身份验证的更多信息，包括如何设置无密码身份验证以及如何在应用程序中构建身份验证流程，请参阅。使用 Amazon Cognito 用户池进行身份验证

以下过程描述了在导入 CSV 文件`RESET_REQUIRED`后，在自定义登录机制中使用本地用户的用户体验。如果您的用户使用托管登录方式登录，请他们选择 “忘记密码？” 选项，提供他们的电子邮件或短信中的代码，然后设置密码。

要求导入的用户重置密码

1. 在您的应用程序中，通过 `InitiateAuth` 使用随机密码以静默方式为当前用户尝试登录。
2. 启用了 `PreventUserExistenceErrors` 时，Amazon Cognito 返回 `NotAuthorizedException`。否则返回 `PasswordResetRequiredException`。
3. 您的应用程序发出 `ForgotPassword` API 请求并重置用户的密码。

a. 应用程序在 `ForgotPassword` API 请求中提交用户名。

b. Amazon Cognito 向经过验证的电子邮件或电话发送代码。目标取决于您在 CSV 文件中为 `email_verified` 和 `phone_number_verified` 提供的值。对 `ForgotPassword` 请求的响应指明了代码的目标。

> **Note**
>
> 必须将您的用户群体配置为验证电子邮件或电话号码。有关更多信息，请参阅 注册并确认用户账户。

c. 您的应用程序向用户显示一条消息，以检查发送代码的位置，并提示用户输入代码和新密码。

d. 用户在应用程序中输入代码和新密码。

e. 应用程序在 `ConfirmForgotPassword` API 请求中提交代码和新密码。

f. 您的应用程序重定向用户以进行登录。

使用用户属性

属性是各种条目的信息，用于帮助您标识单个用户，如名称、电子邮件和电话号码。新的用户池有一组默认标准属性。您也可以在中的用户池定义中添加自定义属性 Amazon Web Services Management Console。本主题将详细介绍这些属性，并为您提供有关如何设置用户池的提示。

请勿将所有与用户相关的信息都应存储在属性中。例如，将频繁变化的用户数据（如使用情况统计数据或游戏分数）保存在单独的数据存储（如 Amazon Cognito Sync 或 Amazon DynamoDB）中。

在将输入的用户属性字符串值提交到用户池之前，请对其进行清理。分析建议的用户属性值的一种方法是使用 Lambda 触发器，例如注册前。

> **Note**
>
> 一些文档和标准将属性称为成员。

主题

- 标准属性
- 用户名和首选用户名
- 自定义登录属性

- 自定义属性
- 属性权限和范围

标准属性

Amazon Cognito 根据 OpenID Connect 规范为所有用户分配一组标准属性。默认情况下，标准和自定义属性值可以是长度不超过 2048 个字符的任何字符串，但是某些属性值有格式限制。

标准属性是：

- `name`
- `family_name`
- `given_name`
- `middle_name`
- `nickname`
- `preferred_username`
- `profile`
- `picture`
- `website`
- `gender`
- `birthdate`
- `zoneinfo`
- `locale`
- `updated_at`
- `address`
- `email`
- `phone_number`
- `sub`

除 `sub` 外，默认情况下，对于所有用户，标准属性都是可选的。要将某个属性设置为必需属性，请在用户池创建过程中，选择属性旁边的 Required（必需）复选框。Amazon Cognito 为每个用户的 `sub` 属性分配一个唯一的用户标识符值。只能验证 email 和 phone_number 属性。

标准属性具有预定义的属性，您可以在 DescribeUserPool API 响应的`SchemaAttributes`参数中查看这些属性。您可以为这些属性设置自定义值，例如数据类型、可变性和长度限制。要修改标准属性属性，请在CreateUserPool 架构参数中设置其自定义值。您还可以在架构中设置必需的属性。在 Amazon Cognito 控制台中创建用户池时，您无法修改标准属性的属性。

> **Note**
>
> 如果您将某个标准属性标记为 Required（必需），则用户必须为该属性提供一个值才能注册。要创建用户而不给出必填属性的值，管理员可以使用 AdminCreateUserAPI。创建用户池后，您无法在必需属性和非必需属性之间切换属性。

标准属性详细信息和格式限制

birthdate

值必须是格式中有效的 10 个字符的日期 YYYY-MM-DD。

电子邮件

用户和管理员可以验证电子邮件地址值。

具有适当 Amazon Web Services 账户 权限的管理员可以更改用户的电子邮件地址，也可以将其标记为已验证。使用 AdminUpdateUserAttributesAPI 或 admin-update-user-attributes Amazon Command Line Interface (Amazon CLI) 命令将电子邮件地址标记为已验证。使用此命令，管理员可以将 `email_verified` 属性更改为 `true`。您也可以在 Amazon Cognito 控制台的用户菜单中编辑用户，将电子邮件地址标记为已验证。

值必须是有效的电子邮件地址字符串，遵循标准电子邮件格式，带有 @ 符号和域名，长度不超过 2048 个字符。

phone_number

如果 SMS 多重验证 (MFA) 处于活动状态，用户必须提供电话号码。有关更多信息，请参阅 向用户池添加 MFA。

用户和管理员可以验证电话号码值。

具有适当 Amazon Web Services 账户 权限的管理员可以更改用户的电话号码，也可以将其标记为已验证。使用 AdminUpdateUserAttributesAPI 或admin-update-user-attributes Amazon CLI 命令将电话号码标记为已验证。使用此命令，管理员可以将 `phone_number_verified` 属性更改

为 `true`。您也可以在 Amazon Cognito 控制台的 “用户” 菜单中编辑用户，将电话号码标记为已验证。

> **Important**
>
> 电话号码必须遵循以下格式规则：电话号码必须以加号 (**+**) 开头，后面紧跟国家/地区代码。电话号码只能包含 **+** 号和数字。先删除电话号码中的任何其他字符，如圆括号、空格或短划线 (**-**)，然后再将该值提交给服务。例如，美国境内的电话号码必须遵循以下格式：**+14325551212**。

preferred_username

您可以将 `preferred_username` 选择为必需或别名，但不能同时选择这两者。如果`preferred_username`是别名，则可以向 UpdateUserAttributesAPI 操作发出请求，并在确认用户后添加属性值。

sub

根据 `sub` 属性对用户编制索引和进行搜索。`sub` 属性是每个用户群体中的唯一用户标识符。用户可以更改如 `phone_number` 和 `email` 等属性。`sub` 属性具有固定的值。有关查找用户的更多信息，请参阅管理和搜索用户账户。

查看必需属性

通过以下过程可查看给定用户池的必需属性。

> **Note**
>
> 在创建用户池后，您无法更改必需属性。

查看必需属性

1. 前往 Amazon Cognito Amazon Web Services Management Console如果控制台提示您，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “注册” 菜单。

5. 在 Required attributes（必需属性）部分中，查看用户池的必需属性。

用户名和首选用户名

`username` 值是一个单独的属性，与 `name` 属性不同。每个用户都有 `username` 属性。Amazon Cognito 会自动为联合用户生成用户名。您必须提供 `username` 属性以在 Amazon Cognito 目录中创建本地用户。创建用户后，您将无法更改 `username` 属性的值。

开发人员可以使用 `preferred_username` 属性为用户提供一个他们可以更改的用户名。有关更多信息，请参阅 自定义登录属性。

如果您的应用程序不需要用户名，就不必要求用户提供用户名。您的应用程序可以在后台为用户创建唯一的用户名。如果您希望用户使用电子邮件地址和密码注册和登录，这非常有用。有关更多信息，请参阅 自定义登录属性。

在用户池中，`username` 必须是唯一的。`username` 可重复使用，但只能是在您已将其删除且不再使用它的情况下。有关属性的字符串限制的信息，请参阅 SignUpAPI 请求的用户名*`username`*属性。

自定义登录属性

创建用户群体时，如果您希望用户能够使用电子邮件地址或电话号码作为其用户名进行注册和登录，则可以设置用户名属性。或者，您可以设置别名属性为用户提供选项：用户可以在注册时包含多个属性，然后使用用户名、首选用户名、电子邮件地址或电话号码登录。

> **Important**
>
> 创建用户池后，您无法更改此设置。

如何在别名属性和用户名属性之间进行选择

您的要求	别名属性	用户名属性
用户有多个登录属性	是[1]	No[2]
用户必须先验证电子邮件地址或电话号码，然后才能使用该地址或电话号码登录	是	否

您的要求	别名属性	用户名属性
使用重复的电子邮件地址或电话号码注册用户并防止出现UsernameExistsException 错误[3]	是	否
可以将相同的电子邮件地址或电话号码属性值分配给多个用户	是[4]	否

[1] 可用的登录属性包括用户名、电子邮件地址、电话号码和首选用户名。

[2] 可以使用电子邮件地址或电话号码进行登录。

[3] 当用户使用可能重复的电子邮件地址或电话号码注册但没有用户名时，您的用户群体不会生成 UsernameExistsException 错误。此行为独立于防止用户名存在错误，此错误适用于登录操作，但不适用于注册操作。

[4] 只有最后验证了该属性的用户才能使用该属性登录。

选项 1：多个登录属性（别名属性）

当用户有用户名但也可以使用该属性登录时，属性就是别名。如果要允许用户在登录表单的用户名字段中选择用户名和其他属性值，请设置别名。该username属性是用户无法更改的固定值。如果您将某个属性标记为别名，用户就可以使用该属性代替用户名来登录。您可以将电子邮件地址、电话号码和首选用户名属性标记为别名。例如，如果您选择电子邮件地址和电话号码作为用户群体的别名，该用户群体中的用户就可以将用户名、电子邮件地址或电话号码与密码一起使用进行登录。

要选择别名属性，请在创建用户群体时选择 User Name（用户名）和至少一个其他登录选项。

> **Note**
>
> 将用户池配置为不区分大小写时，用户可以使用小写或大写字母进行注册或使用别名登录。有关更多信息，请参阅 Amazon Cognito 用户池 API 参考CreateUserPool中的。

如果您选择电子邮件地址作为别名，Amazon Cognito 不接受与有效电子邮件地址格式匹配的用户名。同样，如果您选择电话号码作为别名，Amazon Cognito 将不接受与有效的电话号码格式相匹配的用户群体的用户名。

> **Note**
>
> 在用户池中，别名值必须是唯一的。如果您为电子邮件地址或电话号码配置别名，那么提供的值只能在一个账户中处于已验证状态。在注册期间，如果您的用户提供电子邮件地址或电话号码作为别名值，而另一用户已使用该别名值，注册将成功。然而，当用户尝试使用此电子邮件（或电话号码）确认账户并输入有效的代码时，Amazon Cognito 会返回 `AliasExistsException` 错误。该错误向用户指出，已存在使用此电子邮件地址（或电话号码）的账户。此时，用户可以放弃新账户的创建，并尝试重置旧账户的密码。如果用户继续创建新账户，您的应用程序必须使用 `forceAliasCreation` 选项调用 `ConfirmSignUp` API。`ConfirmSignUp` 和 `forceAliasCreation` 结合会将别名从以前的账户移至新创建的账户，并在以前的账户中将此属性标记为未经验证。

只有在您的用户验证电话号码和电子邮件地址后，电话号码和电子邮件地址才会成为用户的活动别名。如果您将电子邮件地址和电话号码用作别名，我们建议您选择对其进行自动验证。

选择别名属性以防止用户注册时出现电子邮件地址和电话号码属性的 `UsernameExistsException` 错误。

激活 `preferred_username` 属性，以便您的用户可以更改他们用来登录的用户名，而他们的 `username` 属性值不会更改。如果您想设置这种用户体验，请提交新的 `username` 值作为 `preferred_username`，并选择 `preferred_username` 作为别名。这样，用户就可以使用输入的新值登录。如果选择 `preferred_username` 作为别名，您的用户只有在确认账户时才能提供该值。他们在注册期间无法提供该值。

当用户使用用户名注册时，您可以选择他们是否可以使用以下一个或多个别名登录。

- 经过验证的电子邮件地址
- 经过验证的电话号码
- 首选用户名

用户注册后可以更改这些别名。

> **Important**
>
> 当您的用户群体支持使用别名登录，并且您想要向用户授权或查找用户时，请不要通过用户的任何登录属性来识别您的用户。固定值的用户标识符 `sub` 是用户身份的唯一一致指标。

在创建用户池时包括以下步骤，以便用户可以使用别名登录。

Phone number or email address (console)

创建用户池时，必须将电子邮件地址和电话号码设置为别名属性。

在 Amazon Cognito 控制台中创建带有用户名别名的用户池

1. 转到 Amazon Web Services Management Console中的 Amazon Cognito。如果控制台提示您，请输入您的 Amazon 凭据。
2. 使用 “开始使用” 或 “创建用户池” 按钮创建新的用户池。
3. 在 “定义您的应用程序” 中选择应用程序设置。
4. 在 “登录标识符选项” 下的 “配置选项” 中，选中 “用户名” 旁边的复选框以及至少一个其他选项，即电子邮件和电话号码。
5. 选择您的别名属性作为注册的必填属性。在托管登录注册表单中，Amazon Cognito 会提示新用户为必填属性提供值。
6. 在 “添加返回 URL” 下，为托管登录登录后的重定向设置应用程序回调 URL。
7. 选择创建。

Phone number or email address (API/SDK)

使用 CreateUserPoolAPI 操作创建新的用户池。如图所示配置`AliasAttributes`参数。如果您只需要电话号码别名，则可以删除该`email`条目；如果您只需要电子邮件地址别名，则可以删除该`phone_number`条目。

```
"AliasAttributes": [
   "email",
   "phone_number"
],
```

Preferred username (API/SDK)

Amazon Cognito 控制台创建不`preferred_username`使用别名的用户池。要使用`preferred_username`别名创建用户池，请在 Amazon SDK 中设置包含 CreateUserPoolAPI 请求的用户池。要支持在注册时创建首选用户名属性，请`preferred_username`将其设置为必填属性。在托管登录注册表单中，Amazon Cognito 会提示新用户为必填属性提供值。您可以在 Amazon Cognito 控制台中将其设置`preferred_username`为必填属性，但这并不能将其作为别名使用。

配置为别名

preferred_username如图所示，在CreateUserPool请求的AliasAttributes参数中配置为别名。从列表中移除任何你不想作为别名属性的值。

```
"AliasAttributes": [
   "email",
   "phone_number",
   "preferred_username"
],
```

根据需要进行配置

在托管登录注册表单中，Amazon Cognito 会提示新用户为必填属性提供值。preferred_username根据需要在CreateUserPool请求的SchemaAttributes参数中进行配置。

要将首选用户名设置为必填属性，请按如下所示进行配置。以下示例修改了的默认架构preferred_username，使其根据需要进行设置。其他架构参数AttributeDataType（默认为string）和StringAttributeConstraints（长度默认为 1-99 个字符）采用默认值。

```
"Schema": [
   {
      "Name": "preferred_username",
      "Required": true
   }
]
```

选项 2：将电子邮件地址或电话号码作为登录属性（用户名属性）

当用户使用电子邮件地址或电话号码作为其用户名进行注册时，您可以选择他们是否可以仅使用电子邮件地址、仅使用电话号码或其中之一进行注册。

要选择用户名属性，请在创建用户池时不要选择 “用户名” 作为登录选项。

电子邮件地址或电话号码必须是唯一的，并且不能已被其他用户使用。它不必经过验证。用户使用电子邮件地址或电话号码注册之后，将无法使用相同的电子邮件地址或电话号码创建新账户。如果需要，用户只能重复使用现有账户并重置账户的密码。但是，用户可以将电子邮件地址或电话号码更改为新的电子邮件地址或电话号码。如果电子邮件地址或电话号码未被使用，它将成为新的用户名。

当您同时选择电子邮件地址和电话号码作为用户名属性时，用户可以使用其中一个属性登录，即使他们为这两个属性都提供了值。登录用户名基于您在`Username`参数中传递的SignUp值。

> **Note**
>
> 如果用户使用电子邮件地址作为用户名进行注册，则可以将用户名更改为另一个电子邮件地址，但他们无法将用户名更改为电话号码。如果用户使用电话号码进行注册，他们可以将用户名更改为另一个电话号码，但他们无法将用户名更改为电子邮件地址。

在用户池创建过程中使用以下步骤设置使用电子邮件地址或电话号码注册和登录。

Username attributes (console)

以下过程使用电子邮件地址或电话号码用户名属性创建用户池。在 Amazon Cognito 控制台中设置用户名属性的过程的不同之处在于，您不必同时将用户名设置为登录属性。

在 Amazon Cognito 控制台中创建具有用户名属性的用户池

1. 转到 Amazon Web Services Management Console中的 Amazon Cognito。如果控制台提示您，请输入您的 Amazon 凭据。
2. 使用 “开始使用” 或 “创建用户池” 按钮创建新的用户池。
3. 在 “定义您的应用程序” 中选择应用程序设置。
4. 在登录标识符选项下的配置选项中，选择您的用户名属性：电子邮件、电话号码或两者。取消选中用户名。
5. 作为最佳实践，请选择您的用户名属性作为注册的必填属性。在托管登录注册表单中，Amazon Cognito 会提示新用户为必填属性提供值。如果您未按要求设置用户名属性，Amazon Cognito 不会提示新用户为其提供值。在这种情况下，您必须将应用程序配置为收集和提交每个用户的电子邮件地址或电话号码，然后他们才能登录。
6. 在 “添加返回 URL” 下，为托管登录登录后的重定向设置应用程序回调 URL。
7. 选择创建。

Username attributes (API/SDK)

在CreateUserPool请求中，如图所示配置`UsernameAttributes`参数。要仅允许使用电子邮件地址用户名登录，请在此列表中`email`单独指定。要仅允许使用电话号码用户名登录，请单独指定。`phone_number`此参数取代用户名作为登录选项。

```
"UsernameAttributes": [
   "email",
   "phone_number"
],
```

配置用户名属性时，可以发出 SignUpAPI 请求，在username参数中传递电子邮件地址或电话号码。以下是带有用户名属性的代码 SignUp API 操作的行为。

- 例如，如果username字符串采用有效的电子邮件地址格式user@example.com，则用户池会自动使用该username值填充用户的email属性。
- 例如，如果username字符串采用有效的电话号码格式+12065551212，则用户池会自动使用该username值填充用户的phone_number属性。
- 如果 username 字符串格式不是电子邮件或电话号码格式，SignUp API 将返回异常。
- 如果 username 字符串包含已被使用的电子邮件地址或电话号码，SignUp API 将返回异常。
- SignUpAPI 会使用您的用户的 UUID 填充该username属性。此 UUID 与用户身份令牌中的 sub 声明具有相同的值。

APIs除了操作之外，您还可以在所有ListUsers操作中使用电子邮件地址或电话号码代替用户名。在 ListUsers API 请求中，您可以指定Filteremail或phone_number。如果筛选依据username，则必须提供 UUID 用户名，而不是电子邮件地址或电话号码。

自定义属性

您可以将最多 50 个自定义属性添加到您的用户池。您可以为自定义属性指定一个最小和/或最大长度。但是，任何自定义属性的最大长度不能超过 2048 个字符。自定义属性的名称必须与Name参数中描述的正则表达式模式相匹配SchemaAttributeType。

每个自定义属性都具有以下特性：

- 可以将其定义为一个字符串或数字。Amazon Cognito 仅将自定义属性值作为字符串写入 ID 令牌。
- 您不能要求用户为属性提供值。
- 将其添加到用户池后，您将无法删除或更改它。
- 属性名称的字符长度在 Amazon Cognito 接受的限制范围内。有关更多信息，请参阅 Amazon Cognito 中的限额。

- 它可能是可以改变的，也可能是不可改变的。在创建用户时，您只能将值写入不可改变属性。如果您的应用程序客户端具有该属性的写入权限，您可以更改可变属性的值。请参阅属性权限和范围了解更多信息。

> **Note**
>
> 在您的代码和 使用基于角色的访问控制 的规则设置中，自定义属性需要使用 `custom:` 前缀，以便将它们与标准属性区分开来。

在创建用户池时，您还可以在的属性中添加开发者属性CreateUserPool。`SchemaAttributes`开发人员属性具有 `dev:` 前缀。您只能使用 Amazon 凭证修改用户的开发者属性。开发人员属性是一项旧版特征，Amazon Cognito 已将其替换为应用程序客户端读写权限。

通过以下过程创建新的自定义属性。

使用控制台添加自定义属性

1. 前往 Amazon Cognito Amazon Web Services Management Console如果控制台提示您，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “注册” 菜单，然后在 “自定义属性” 部分中，选择 “添加自定义属性”。
5. 在存储库的 Add custom attributes（添加自定义属性）页面上，提供有关新属性的以下详细信息：
   - 输入 Name（名称）。
   - 选择 Type（类型）（字符串或数字）。
   - 输入最小字符串长度或数字值。
   - 输入最大字符串长度或数字值。
   - 如果您想授予用户在设置初始值后更改自定义属性值的权限，请选择 Mutable（可变）。
6. 选择 Save changes（保存更改）。

属性权限和范围

对于每个应用程序客户端，您可以为每个用户属性设置读取和写入权限。这样，您可以控制为了读取和修改您为用户存储的每个属性，任何应用程序所具有的访问权限。例如，您可以设置一个自定义属性，

用于指明用户是否为付费客户。您的应用程序可能能够看到此属性，但无法直接更改它。相反，您可以使用管理工具或后台进程更新此属性。您可以通过 Amazon Cognito 控制台、Amazon Cognito API 或 Amazon CLI设置用户属性的权限。默认情况下，任何新的自定义属性都不可用，直到您为其设置读取和写入权限。默认情况下，创建新的应用程序客户端时，您将授予应用程序对所有标准和自定义属性的读取和写入权限。要将应用程序限制为仅使用它所需的信息量，请在应用程序客户端配置中为属性分配特定权限。

妥善做法是在创建应用程序客户端时指定属性的读取和写入权限。向您的应用程序客户端授予应用程序运行所需的最小用户属性集的访问权限。

> **Note**
>
> DescribeUserPoolClient只有在配置默认权限之外的应用程序客户端权限`WriteAttributes`时，才会返回`ReadAttributes`和的值。

更新属性权限（Amazon Web Services Management Console）

1. 前往 Amazon Cognito Amazon Web Services Management Console如果控制台提示您，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “应用程序客户端” 菜单，然后从列表中选择一个应用程序客户端。
5. 在 “属性权限” 选项卡中，选择 “编辑”。
6. 在 Edit attribute read and write permissions（编辑设置属性读取和写入权限）页面上，配置读取和写入权限，然后选择 Save changes（保存更改）。

使用自定义属性对每个应用程序客户端重复这些步骤。

对于每个应用程序客户端，您可以将属性标记为可读或可写。这对于标准属性和自定义属性均适用。您的应用程序可以检索您标记为可读的属性的值，也可以设置或修改您标记为可写的属性的值。如果您的应用程序尝试为其无权写入的属性设置值，Amazon Cognito 会返回。`NotAuthorizedException` GetUser请求包括带有应用程序客户端声明的访问令牌；Amazon Cognito 仅返回应用程序客户端可以读取的属性的值。来自应用程序的用户 ID 令牌仅包含与可读属性相对应的声明。所有应用程序客户端都可以写入用户群体所需的属性。仅当您还为尚未具有值的必需属性提供值时，才能在 Amazon Cognito 用户群体 API 请求中设置相应属性的值。

自定义属性具有不同的读写权限特征。您可以将它们创建为用户群体的可变或不可变属性，并可以将它们设置为任何应用程序客户端的读或写属性。

在创建用户期间，不可变的自定义属性可以更新一次。您可以使用以下方法填充不可变的属性。

- `SignUp`：用户注册了一个应用程序客户端，该客户端具有对不可变自定义属性的写访问权限。它们为该属性提供了一个值。
- 使用第三方 IdP 登录：用户登录到一个应用程序客户端，该客户端具有对不可变自定义属性的写访问权限。IdP 的用户群体配置有一条规则，可以将提供的声明映射到不可变的属性。这可以实现，但不太实际，因为用户只能登录一次。在首次登录后，如果用户尝试再次登录，Amazon Cognito 会因为映射规则指向一个现在不可写的属性而拒绝登录尝试。
- `AdminCreateUser`：您为不可变属性提供一个值。

使用范围的属性权限

在使用 Amazon SDK 或 CDK、REST API 或配置的用户池中 Amazon CLI，您可以使用 OIDC 范围配置应用程序客户端的读取或写入权限。`oidc:profileoidc:profile` 授予对以下标准属性的读取或写入权限：

- `name`
- `family_name`
- `given_name`
- `middle_name`
- `nickname`
- `preferred_username`
- `profile`
- `picture`
- `website`
- `gender`
- `birthdate`
- `zoneinfo`
- `locale`

此列表是 OIDC 标准属性减去 email、phone_number、sub 和 address，如 OIDC 规范第 2.4 节中所定义。有关您可以分配给应用程序客户端的范围的信息，请参阅作用域、M2M 以及 APIs 带资源服务器的场景。

要将您的应用程序客户端配置为写入oidc:profile作用域下的属性，请在CreateUserPoolClient或UpdateUserPoolClientAPI 请求中WriteAttributes将oidc:profile的值设置为，以及您希望允许应用程序修改的任何其他属性。同样，要授予对这些属性的读取权限，oidc:profile请添加的值ReadAttributes。

您可以在创建用户池后更改属性权限和范围。

了解用户池 JSON 网络令牌 (JWTs)

令牌是身份验证的工件，您的应用程序可以将其用作 OIDC 身份验证的证明和请求对资源的访问权限。令牌中的声明是有关您的用户的信息。ID 令牌包含有关用户身份的声明，例如他们的用户名、姓氏和电子邮件地址。访问令牌包含诸如经过身份验证scope的用户可用于访问第三方 APIs、Amazon Cognito 用户自助服务 API 操作的声明，以及. userInfo 端点 访问令牌和 ID 令牌均包含 cognito:groups 声明，其中包含用户在用户群体中的组成员资格。有关用户群体组的更多信息，请参阅 向用户池添加组。

Amazon Cognito 还有刷新令牌，您可以使用它来获取新的令牌或撤销现有令牌。刷新令牌来检索新的 ID 令牌和访问令牌。撤消令牌来撤消刷新令牌允许的用户访问权限。

亚马逊 Cognito 以 base64 url 编码的字符串的形式发行代币。您可以将任何 Amazon Cognito ID 或访问令牌base64url解码为纯文本 JSON。Amazon Cognito 刷新令牌已加密，对用户群体用户和管理员不透明，并且只能由您的用户群体读取。

使用令牌进行身份验证

当用户登录您的应用程序时，Amazon Cognito 会验证登录信息。如果登录成功，Amazon Cognito 会创建会话并为经过身份验证的用户返回 ID 令牌、访问令牌和刷新令牌。您可以使用这些令牌向您的用户授予对下游资源的访问权限， APIs 例如 Amazon API Gateway。或者，您可以用它们交换用于访问其他 Amazon Web Services 服务的临时 Amazon 凭证。

存储令牌

您的应用程序必须能够存储不同大小的令牌。令牌大小变化的原因可能包括但不限于其它声明、编码算法的更改以及加密算法的更改等。当您在用户群体中启用令牌吊销时，Amazon Cognito 会向 JSON Web 令牌添加其他声明，从而增加令牌大小。新 `origin_jti` 和 `jti` 声明已添加到访问和 ID 令牌中。有关令牌撤消的更多信息，请参阅撤消令牌。

> **Important**
>
> 最佳实践是在应用程序环境中保护传输和存储中的所有令牌。令牌可以包含有关用户的个人识别信息，以及有关用于用户池的安全模型的信息。

自定义令牌

您可以自定义 Amazon Cognito 传递给应用程序的访问令牌和 ID 令牌。在 令牌生成前 Lambda 触发器中，您可以添加、修改和隐藏令牌声明。令牌生成前触发器是一个 Lambda 函数，Amazon Cognito 会向其发送一组默认声明。声明包括 OAuth 2.0 范围、用户池群组成员资格、用户属性等。然后，该函数可以根据这些信息在运行时进行更改，并将更新的令牌声明返回给 Amazon Cognito。

使用版本 2 事件自定义访问令牌需要支付额外费用。有关更多信息，请参阅 Amazon Cognito 定价。

主题

- 了解身份（ID）令牌
- 了解访问令牌
- 刷新令牌
- 通过令牌撤销来结束用户会话
- 验证 JSON 网络令牌
- 管理用户池令牌到期和缓存

了解身份（ID）令牌

ID 令牌是一个 JSON Web 令牌（JWT），其中包含有关经身份验证的用户的身份声明，如 `name`、`email` 和 `phone_number`。您可以在应用程序中使用此身份信息。此外，ID 令牌还可用于针对资源服务器或服务器应用程序对用户进行身份验证。您还可以将应用程序外部的 ID 令牌用于 Web API 操作。在这些情况下，您必须先验证 ID 令牌的签名，然后才能信任 ID 令牌内的任何声明。请参阅验证 JSON 网络令牌。

您可以将 ID 令牌过期时间设置为 5 分钟到 1 天之间的任何值。您可以按应用程序客户端设置此值。

> **Important**
>
> 当您的用户使用托管登录方式登录时，Amazon Cognito 会设置有效期为 1 小时的会话 Cookie。如果您在应用程序中使用托管登录进行身份验证，并将访问权限和 ID 令牌的最短持续时间指定为 1 小时以内，则在 Cookie 过期之前，您的用户仍将拥有有效的会话。如果用户的令牌在一小时的会话期间过期，则用户可以刷新他们的令牌，而无需重新身份验证。

ID 令牌标头

标头包含两部分信息：密钥 ID (`kid`) 和算法 (`alg`)。

```
{
"kid" : "1234example=",
"alg" : "RS256"
}
```

`kid`

密钥 ID。其值指示用于保护令牌的 JSON Web Signature (JWS) 的密钥。您可以在`jwks_uri`终端节点上查看您的用户池签 IDs 名密钥。

有关 `kid` 参数的更多信息，请参阅密钥标识符 (kid) 标头参数。

`alg`

Amazon Cognito 用于保护访问令牌的加密算法。用户池使用 RS256 加密算法，即带有 SHA-256 的 RSA 签名。

有关 `alg` 参数的更多信息，请参阅算法（alg）标头参数。

ID 令牌默认有效载荷

这是来自 ID 令牌的示例有效载荷。它包含有关经过身份验证的用户的声明。有关 OpenID Connect（OIDC）标准声明的更多信息，请参阅 OIDC 标准声明列表。您可以使用 令牌生成前 Lambda 触发器 添加自有设计的声明。

```
<header>.{
    "sub": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:groups": [
```

```
        "test-group-a",
        "test-group-b",
        "test-group-c"
    ],
    "email_verified": true,
    "cognito:preferred_role": "arn:aws:iam::111122223333:role/my-test-role",
    "iss": "https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
    "cognito:username": "my-test-user",
    "middle_name": "Jane",
    "nonce": "abcdefg",
    "origin_jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "cognito:roles": [
        "arn:aws:iam::111122223333:role/my-test-role"
    ],
    "aud": "xxxxxxxxxxxxexample",
    "identities": [
        {
            "userId": "amzn1.account.EXAMPLE",
            "providerName": "LoginWithAmazon",
            "providerType": "LoginWithAmazon",
            "issuer": null,
            "primary": "true",
            "dateCreated": "1642699117273"
        }
    ],
    "event_id": "64f513be-32db-42b0-b78e-b02127b4f463",
    "token_use": "id",
    "auth_time": 1676312777,
    "exp": 1676316377,
    "iat": 1676312777,
    "jti": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
    "email": "my-test-user@example.com"
}
.<token signature>
```

sub

经过身份验证的用户的唯一标识符（UUID）或主题。用户名在您的用户群体中可能不是唯一的。sub 声明是识别给定用户的最佳方法。

cognito:groups

以您的用户为成员的用户群体组的名称数组。组可以是您提供给应用程序的标识符，也可以从身份池为首选 IAM 角色生成请求。

cognito:preferred_role

与用户的最高优先级用户群体组关联的 IAM 角色的 ARN。有关您的用户群体如何选择此角色声明的更多信息，请参阅将优先级值分配到组。

iss

发行令牌的身份提供者。声明采用以下格式。

```
https://cognito-idp.<Region>.amazonaws.com/<your user pool ID>
```

cognito:username

用户群体中用户的用户名。

nonce

nonce声明来自同名参数，您可以将其添加到对 OAuth 2.0 authorize 终端节点的请求中。添加参数时，nonce 声明包含在 Amazon Cognito 颁发的 ID 令牌中，您可以使用它来防范重播攻击。如果在您的请求中未提供 nonce 值，当您通过第三方身份提供商进行身份验证时，Amazon Cognito 会自动生成并验证随机数，然后将其作为 nonce 声明添加到 ID 令牌中。Amazon Cognito 中的 nonce 声明的实现基于 OIDC 标准。

origin_jti

与用户的刷新令牌关联的令牌撤消标识符。Amazon Cognito 在检查您是否通过撤消端点或 API 操作撤销了用户的令牌时会引用该origin_jti声明。RevokeToken当您撤消令牌时，Amazon Cognito 会使所有具有相同 origin_jti 值的访问令牌和 ID 令牌失效。

cognito:roles

与您的用户组关联的 IAM 角色的名称的数组。每个用户群体组可以有一个与之关联的 IAM 角色。此数组显示了您的用户组的所有 IAM 角色，不按优先级排列。有关更多信息，请参阅 向用户池添加组。

aud

对用户进行身份验证的用户群体应用程序客户端。Amazon Cognito 在访问令牌 client_id 声明中呈现相同的值。

identities

用户的 identities 属性的内容。该属性包含有关您通过联合登录或通过将联合用户与本地配置文件关联而与用户关联的每个第三方身份提供者配置文件的信息。此信息包含其提供者名称、提供者唯一 ID 和其它元数据。

token_use

令牌的预期用途。在 ID 令牌中，其值为 `id`。

auth_time

您的用户完成身份验证的身份验证时间，采用 Unix 时间格式。

exp

您的用户令牌的过期时间，采用 Unix 时间格式。

iat

Amazon Cognito 颁发您的用户令牌的时间，采用 Unix 时间格式。

jti

JWT 的唯一标识符。

ID 令牌可包含 OIDC 标准声明中所定义的 OIDC 标准声明。ID 令牌还可包含您在用户池中定义的自定义属性。无论属性类型如何，Amazon Cognito 都会将自定义属性值作为字符串写入 ID 令牌。

> **Note**
>
> 用户池自定义属性始终以 `custom:` 为前缀。

ID 令牌签名

ID 令牌的签名根据 JWT 令牌的标头和负载计算。在您接受应用程序收到的任何 ID 令牌中的声明之前，请验证该令牌的签名。有关更多信息，请参阅“验证 JSON Web 令牌”。验证 JSON 网络令牌。

了解访问令牌

用户池访问令牌包含有关经身份验证的用户声明、用户组列表以及范围列表。访问令牌的目的是授权执行 API 操作。您的用户群体接受访问令牌以授权用户执行自助操作。例如，您可以使用访问令牌授予用户访问权限以添加、更改或删除用户属性。

通过访问令牌中的 OAuth 2.0 作用域（源自您添加到用户池中的自定义范围），您可以授权您的用户从 API 检索信息。例如，Amazon API Gateway 支持使用 Amazon Cognito 访问令牌进行授权。您可以使用用户群体中的信息填充 REST API 授权方，也可以使用 Amazon Cognito 作为 HTTP API 的 JSON 网络令牌（JWT）授权方。要生成具有自定义范围的访问令牌，您必须通过用户群体公有端点请求访问令牌。

借助 Essentials 或 Plus 功能计划，您还可以实现令牌生成前 Lambda 触发器，在运行时为您的访问令牌添加范围。有关更多信息，请参阅 令牌生成前 Lambda 触发器。

具有openid作用域的用户的访问令牌是可以从中请求有关您的用户属性的更多信息的权限userInfo端点。来自userInfo端点的信息量来自访问令牌中的其他范围：例如，profile对于所有用户数据，email针对他们的电子邮件地址。

用户的访问令牌的aws.cognito.signin.user.admin范围是读取和写入用户属性、列出身份验证因素、配置多重身份验证 (MFA) 首选项以及管理记住的设备的权限。您的访问令牌授予此范围的属性访问权限级别与您分配给应用程序客户端的属性读/写权限相匹配。

访问令牌是 JSON Web 令牌 (JWT)。访问令牌的标头与 ID 令牌具有相同的结构。Amazon Cognito 使用与签署 ID 令牌的密钥所不同的密钥对访问令牌进行签名。访问密钥 ID (kid) 声明的值与来自同一用户会话的 ID 令牌中 kid 声明的值不匹配。在您的应用程序代码中，单独验证 ID 令牌和访问令牌。在验证签名之前，请勿信任访问令牌中的声明。有关更多信息，请参阅 验证 JSON 网络令牌。您可以将访问令牌过期时间设置为 5 分钟到 1 天之间的任何值。您可以按应用程序客户端设置此值。

> **Important**
>
> 对于访问权限和 ID 令牌，如果您使用托管登录，请不要指定至少少于一小时的时间。托管登录会设置有效期为一小时的浏览器 Cookie。如果您将访问令牌的持续时间配置为少于一小时，则这不会影响托管登录 Cookie 的有效性，也不会影响用户在初始登录后一小时内无需其他凭据即可重新进行身份验证的能力。

访问令牌标头

标头包含两部分信息：密钥 ID (kid) 和算法 (alg)。

```
{
"kid" : "1234example="
"alg" : "RS256",
}
```

kid

密钥 ID。其值指示用于保护令牌的 JSON Web Signature (JWS) 的密钥。您可以在jwks_uri终端节点上查看您的用户池签 IDs 名密钥。

有关 kid 参数的更多信息，请参阅密钥标识符 (kid) 标头参数。

alg

Amazon Cognito 用于保护访问令牌的加密算法。用户池使用 RS256 加密算法，即带有 SHA-256 的 RSA 签名。

有关 alg 参数的更多信息，请参阅算法（alg）标头参数。

访问令牌默认有效载荷

这是来自访问令牌的示例负载。有关更多信息，请参阅 JWT 声明。您可以使用 令牌生成前 Lambda 触发器 添加自有设计的声明。

```
<header>.
{
   "sub":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "device_key": "aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "cognito:groups":[
      "testgroup"
   ],
   "iss":"https://cognito-idp.us-west-2.amazonaws.com/us-west-2_example",
   "version":2,
   "client_id":"xxxxxxxxxxxxexample",
   "origin_jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "event_id":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "token_use":"access",
   "scope":"phone openid profile resourceserver.1/appclient2 email",
   "auth_time":1676313851,
   "exp":1676317451,
   "iat":1676313851,
   "jti":"aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee",
   "username":"my-test-user"
}
.<token signature>
```

sub

经过身份验证的用户的唯一标识符（UUID）或主题。用户名在您的用户群体中可能不是唯一的。sub 声明是识别给定用户的最佳方法。

cognito:groups

以您的用户为成员的用户群体组的名称数组。

iss

发行令牌的身份提供者。声明采用以下格式。

`https://cognito-idp.us-east-1.amazonaws.com/us-east-1_EXAMPLE`

client_id

对用户进行身份验证的用户群体应用程序客户端。Amazon Cognito 在 ID 令牌 aud 声明中呈现相同的值。

origin_jti

与用户的刷新令牌关联的令牌撤消标识符。Amazon Cognito 在检查您是否通过撤消端点或 API 操作撤销了用户的令牌时会引用该`origin_jti`声明。RevokeToken当您撤销令牌时，Amazon Cognito 将不再验证具有相同值的访问令牌和身份令牌。`origin_jti`

token_use

令牌的预期用途。在访问令牌中，其值为 `access`。

scope

向登录用户发布的 OAuth 2.0 范围列表。作用域定义令牌为`userInfo`端点上的外部 APIs、用户自助服务操作和用户数据提供的访问权限。令牌端点 中的令牌可以包含您的应用程序客户端支持的任何范围。来自 Amazon Cognito API 登录的令牌仅包含范围 `aws.cognito.signin.user.admin`。

auth_time

您的用户完成身份验证的身份验证时间，采用 Unix 时间格式。

exp

您的用户令牌的过期时间，采用 Unix 时间格式。

iat

Amazon Cognito 颁发您的用户令牌的时间，采用 Unix 时间格式。

jti

JWT 的唯一标识符。

username

用户池中用户的用户名。

更多资源

- 如何在 Amazon Cognito 用户池中自定义访问令牌

访问令牌签名

访问令牌的签名使用在`.well-known/jwks.json`端点发布的密钥进行签名，可验证令牌标头和有效负载的完整性。当您使用访问令牌授权外部访问时 APIs，请务必将您的 API 授权方配置为根据签名该签名的密钥来验证此签名。有关更多信息，请参阅 验证 JSON 网络令牌。

刷新令牌

您可以使用刷新令牌来检索新的 ID 令牌和访问令牌。默认情况下，刷新令牌会在您的应用程序用户登录用户池的 30 天后过期。当您为用户池创建应用程序时，您可以将应用程序的刷新令牌到期时间设置为介于 60 分钟和 10 年之间的任何值。

使用刷新令牌获取新的访问权限和身份令牌

Amazon Cognito 会发出刷新令牌，以响应通过托管登录授权代码流程和 API 操作或 SDK 方法成功进行身份验证。刷新令牌会返回新的 ID 和访问令牌，还可以返回新的刷新令牌（可选）。您可以通过以下方式使用刷新令牌。

GetTokensFromRefreshToken

GetTokensFromRefreshTokenAPI 操作从有效的刷新令牌中发布新的 ID 和访问令牌。如果您启用了刷新令牌轮换，则还会获得新的刷新令牌。

InitiateAuth 和 AdminitiateAuth

AdminInitiateAuth或 InitiateAuthAPI 操作包括`REFRESH_TOKEN_AUTH`身份验证流程。在此流程中，您将传递刷新令牌并获得新的 ID 和访问令牌。启用刷新令牌轮换后，您无法使用应用程序`REFRESH_TOKEN_AUTH`内客户端进行身份验证。

OAuth 令牌端点

具有域的用户池中的令牌端点具有一种`refresh_token`授予类型，用于从有效的刷新令牌中颁发新 ID、访问权限以及可选（使用刷新令牌轮换）刷新令牌。

刷新代币轮换

通过刷新令牌轮换，您可以选择将用户池配置为使原始刷新令牌失效，并在每次刷新令牌时发出新的刷新令牌。启用此设置后，所有形式的令牌刷新成功请求都会返回一个新的 ID、访问权限和刷新令牌。新的刷新令牌在原始刷新令牌的剩余时间内有效。您可以将应用程序客户端配置为轮换刷新令牌或延续原始刷新令牌。要允许在短时间内重试，您还可以为原始刷新令牌配置最多 60 秒的宽限期。

关于刷新令牌轮换的注意事项

- 启用刷新令牌轮换后，您的用户池中的 JSON Web 令牌中会添加新的声明。`origin_jti` 和 `jti` 声明已添加到访问和 ID 令牌中。这些索赔增加了索赔的规模 JWTs。
- 刷新令牌轮换与身份验证流程不兼容`REFRESH_TOKEN_AUTH`。要实现刷新令牌轮换，您必须在应用程序客户端中禁用此身份验证流程，并将您的应用程序设计为使用 GetTokensFromRefreshTokenAPI 操作或等效的 SDK 方法提交令牌刷新请求。
- 当刷新令牌轮换处于非活动状态时，您可以使用或完成令牌刷新请求。`GetTokensFromRefreshToken` `REFRESH_TOKEN_AUTH`
- 当设备记忆在用户池中处于活动状态时，您必须在`GetTokensFromRefreshToken`请求中提供设备密钥。如果您的用户没有您的应用程序在初始身份验证请求中提交的已确认设备密钥，Amazon Cognito 会发布一个新的密钥。要刷新此配置中的令牌，您必须提供设备密钥，无论您是在身份验证响应中`AuthParameters`指定了设备密钥还是在身份验证响应中收到了新密钥。
- 您可以在请求中传递`ClientMetadata`到令牌生成前 Lambda 触发器。`GetTokensFromRefreshToken`这些数据将传递给触发器的输入事件，并提供其他上下文，您可以在 Lambda 函数的自定义逻辑中使用这些上下文。

作为安全最佳实践，请在应用程序客户端上启用刷新令牌轮换。

Enable refresh token rotation (console)

以下过程打开或关闭应用程序客户端的刷新令牌轮换。此过程需要现有的应用程序客户端。要了解有关创建应用程序客户端的更多信息，请参阅特定于应用程序的应用程序客户端设置。

启用刷新令牌轮换

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 导航到应用程序客户端菜单，然后选择现有的应用程序客户端。

5. 从页面的 “应用程序客户端信息” 部分选择 “编辑”。
6. 在 “高级安全配置” 下，找到 “启用刷新令牌轮换” 选项。
7. 要启用旋转，请选中该复选框。要禁用旋转，请取消选中该复选框。
8. 在刷新令牌轮换宽限期下，输入要设置为撤消轮换刷新令牌之前的延迟时间（最多 60 秒）。

Enable refresh token rotation (API)

在CreateUserPoolClient或 UpdateUserPoolClientAPI 请求中配置刷新令牌轮换。以下部分请求正文开启刷新令牌轮换并将宽限期设置为十秒。

```
"RefreshTokenRotation" : {
   "Feature" : "ENABLED,
   "RetryGracePeriodSeconds" : 10
}
```

API 和 SDK 令牌刷新

有两种方法可以使用刷新令牌通过用户池 API 获取新 ID 和访问令牌，具体取决于刷新令牌轮换是否处于活动状态。在启用刷新令牌轮换的应用程序客户端中，使用 GetTokensFromRefreshTokenAPI 操作。在没有刷新令牌轮换的应用程序客户端中，使用AdminInitiateAuth或 InitiateAuthAPI 操作`REFRESH_TOKEN_AUTH`流程。

> **Note**
>
> 用户可以在托管登录或使用您构建的自定义应用程序 Amazon SDKs 和 Amazon Cognito API 操作中使用用户池进行身份验证。`REFRESH_TOKEN_AUTH`流程和`GetTokensFromRefreshToken`都可以为托管登录用户完成令牌刷新。自定义应用程序中的令牌刷新不会影响托管登录会话。这些会话在浏览器 Cookie 中设置，有效期为一小时。`GetTokensFromRefreshToken`响应会发出新的 ID、访问权限和可选的刷新令牌，但不会续订托管登录会话 Cookie。
> `REFRESH_TOKEN_AUTH`在启用刷新令牌轮换的应用程序客户端中不可用。

GetTokensFromRefreshToken

GetTokensFromRefreshToken返回您使用刷新令牌授权的请求中的新 ID、访问和刷新令牌。以下是的请求正文示例GetTokensFromRefreshToken。您可以在请求此操作时向 Lambda 触发器提交客户端元数据。

```
{
    "RefreshToken": "eyJjd123abcEXAMPLE",
    "ClientId": "1example23456789",
    "ClientSecret": "myappclientsecret123abc",
    "ClientMetadata": {
      "MyMetadataKey" : "MyMetadataValue"
   },
}
```

AdminInitiateAuth/InitiateAuth

要在刷新令牌轮换处于非活动状态时使用刷新令牌，请使用AdminInitiateAuth或 InitiateAuthAPI 操作。为 AuthFlow 参数传递 REFRESH_TOKEN_AUTH。在 AuthFlow 的 AuthParameters 属性中，将用户的刷新令牌作为 "REFRESH_TOKEN" 的值进行传递。在您的 API 请求通过所有质询后，Amazon Cognito 会返回新的 ID 和访问令牌。

以下是使用InitiateAuth或 AdminInitiateAuth API 刷新令牌的请求正文示例。

```
{
    "AuthFlow": "REFRESH_TOKEN_AUTH",
    "ClientId": "1example23456789",
    "UserPoolId": "us-west-2_EXAMPLE",
    "AuthParameters": {
        "REFRESH_TOKEN": "eyJjd123abcEXAMPLE",
        "SECRET_HASH": "kT5acwCVrbD6JexhW3EQwnRSe6fLuPTRkEQ50athqv8="
    }
}
```

OAuth 令牌刷新

您也可以将刷新令牌提交给用户群体中您在其中配置了域的令牌端点。在请求正文中，加入 refresh_token 的 grant_type 值和用户刷新令牌的 refresh_token 值。

对令牌端点的请求可在启用刷新令牌轮换的应用程序客户端和处于非活动状态的应用程序客户端中使用。当刷新令牌轮换处于活动状态时，令牌端点会返回一个新的刷新令牌。

以下是带有刷新令牌的请求示例。

```
POST /oauth2/token HTTP/1.1
Host: auth.example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic ZGpj0Th1M2ppZWRtaTI4M2V10TI40mFiY2RlZjAxMjM0NTY30Dkw
Content-Length: **

client_id=1example23456789&grant_type=refresh_token&refresh_token=eyJjd123abcEXAMPLE
```

撤消刷新令牌

您可以撤销属于用户的刷新令牌。有关撤消令牌的更多信息，请参阅通过令牌撤销来结束用户会话。

> **Note**
>
> 撤消刷新令牌将撤消 Amazon Cognito 从具有该令牌的刷新请求中发出的所有 ID 和访问令牌。

要让用户退出当前所有登录会话，请使用或 API 请求撤消其所有令牌。GlobalSignOutAdminUserGlobalSignOut注销用户后，会发生以下影响。

- 用户的刷新令牌无法获取该用户的新令牌。
- 用户的访问令牌无法发出经过令牌授权的 API 请求。
- 用户必须重新进行身份验证以获取新的令牌。由于托管登录会话 Cookie 不会自动过期，因此您的用户可以使用会话 Cookie 重新进行身份验证，无需额外提示输入凭据。在您注销托管登录用户后，将他们重定向到，Amazon Cognito 将在那里清除他们的会话 cookie。注销端点

使用刷新令牌，您可以将用户的会话长时间保留在您的应用程序中。随着时间的推移，您的用户可能希望取消对某些使用刷新令牌保持登录状态的应用程序的授权。要将您的用户从单个会话中注销，请撤消他们的刷新令牌。当您的用户想要退出所有经过身份验证的会话时，请生成 GlobalSignOutAPI 请求。应用程序可以为用户提供一个选择，如从所有设备注销。`GlobalSignOut` 接受用户的有效（即未更改、未过期、未撤消的）访问令牌。由于此 API 经过令牌授权，因此一个用户无法使用它来发起另一个用户的注销。

但是，您可以生成一个 AdminUserGlobalSignOutAPI 请求，该请求由您使用您的 Amazon 凭据进行授权，以便将任何用户从其所有设备上注销。管理员应用程序必须使用 Amazon 开发者凭据调用此 API 操作，并将用户池 ID 和用户名作为参数传递。`AdminUserGlobalSignOut` API 可以在用户池中注销任何用户。

有关您可以使用 Amazon 凭证或用户访问令牌授权的请求的更多信息，请参阅按授权模型分组的 API 操作列表。

通过令牌撤销来结束用户会话

您可以使用以下方法撤消刷新令牌和最终用户会话。撤消刷新令牌后，先前由该刷新令牌颁发的所有访问令牌都将无效。向用户颁发的其他刷新令牌不受影响。

RevokeToken 操作

RevokeToken撤消给定刷新令牌的所有访问令牌，包括来自交互式登录的初始访问令牌。此操作不会影响用户的任何其他刷新令牌或这些其他刷新令牌的 ID 和访问令牌子级。

撤销端点

撤销端点会撤消给定的刷新令牌以及刷新令牌生成的所有 ID 和访问令牌。此端点还会撤消交互式登录的初始访问令牌。对该端点的请求不会影响用户的任何其他刷新令牌或这些其他刷新令牌的 ID 和访问令牌子级。

GlobalSignOut 操作

GlobalSignOut是一种自助操作，用户使用其访问令牌对其进行授权。此操作会撤消请求用户的所有刷新、ID 和访问令牌。

AdminUserGlobalSignOut 操作

AdminUserGlobalSignOut是管理员使用 IAM 凭证授权的服务器端操作。此操作会撤消目标用户的所有刷新、ID 和访问令牌。

关于撤销代币的注意事项

- 您撤消刷新令牌的请求必须包括用于获取令牌的客户端 ID。
- 用户池 JWTs 是独立的，其签名和到期时间是在创建令牌时分配的。已撤消令牌无法与任何需要令牌的 Amazon Cognito API 调用一起使用。但是，如果使用任何验证令牌签名和过期的 JWT 库进行验证，已撤消令牌仍然有效。
- 当您创建新的用户池客户端时，默认会启用令牌撤消。
- 您只能在启用了令牌撤消功能的应用程序客户端中撤消刷新令牌。
- 启用令牌吊销后，Amazon Cognito JSON Web 令牌中会添加新的声明。`origin_jti` 和 `jti` 声明已添加到访问和 ID 令牌中。这些声明增加应用程序客户端访问和 ID 令牌的大小。
- 当您在之前启用令牌吊销的应用程序客户端中禁用令牌撤销时，已撤销的令牌不会再次处于活动状态。

- 当您禁用用户帐户（撤消刷新和访问令牌）时，如果您再次启用该用户帐户，则已撤销的令牌不会变为活动状态。
- 使用 Amazon Web Services Management Console、或 Amazon API 创建新的用户池客户端时 Amazon CLI，默认情况下会启用令牌撤销。

启用令牌撤消

在您为现有的用户池客户端撤消令牌前，必须启用令牌撤销。您可以使用 Amazon CLI 或 Amazon API 为现有用户池客户端启用令牌撤销。为此，请调用 `aws cognito-idp describe-user-pool-client` CLI 命令或 `DescribeUserPoolClient` API 操作以从应用程序客户端检索当前设置。然后调用 `aws cognito-idp update-user-pool-client` CLI 命令或 `UpdateUserPoolClient` API 操作。包括来自您的应用客户端的当前设置并将 `EnableTokenRevocation` 参数设置为 `true`。

要使用 Amazon Cognito API 或软件开发工具包创建或修改启用令牌撤销功能的 Amazon 应用程序客户端，请在CreateUserPoolClient您的UpdateUserPoolClient或 API 请求中包含以下参数。

```
"EnableTokenRevocation": true
```

要在 Amazon Cognito 控制台中配置令牌撤销，请从用户池的应用程序客户端菜单中选择一个应用程序客户端。在 “应用程序客户端信息” 中选择 “编辑” 按钮，然后在 “高级配置” 下启用或禁用令牌撤销。

撤消令牌

您可以使用 RevokeTokenAPI 请求撤消刷新令牌，例如使用 `aws cognito-idp revoke-token` CLI 命令。您也可以使用撤消端点撤消令牌。此端点在您将域添加到用户池后可用。您可以在 Amazon Cognito 托管域或您的自定义域上使用撤消端点。

以下是示例 `RevokeToken` API 请求的正文。

```
{
   "ClientId": "1example23456789",
   "ClientSecret": "abcdef123456789ghijklexample",
   "Token": "eyJjdHkiOiJKV1QiEXAMPLE"
}
```

以下是对使用自定义域的用户群体的 `/oauth2/revoke` 端点发出的 cURL 请求示例。

```
curl --location 'auth.mydomain.com/oauth2/revoke' \
```

```
--header 'Content-Type: application/x-www-form-urlencoded' \
--header 'Authorization: Basic Base64Encode(client_id:client_secret)' \
--data-urlencode 'token=abcdef123456789ghijklexample' \
--data-urlencode 'client_id=1example23456789'
```

除非您的应用程序客户端具有客户端密钥，否则 `RevokeToken` 操作和 `/oauth2/revoke` 端点不需要额外授权。

验证 JSON 网络令牌

JSON 网络令牌 (JWTs) 可以轻松解码、读取和修改。修改过的访问令牌会带来权限提升的风险。修改过的 ID 令牌会带来冒充身份的风险。您的应用程序信任您的用户池作为令牌颁发者，但如果用户拦截了传输中的令牌，该怎么办？ 您必须确保应用程序收到的令牌与 Amazon Cognito 发放的令牌一致。

Amazon Cognito 发放令牌，这些令牌使用 OpenID Connect（OIDC）规范的某些完整性和保密性特征。用户池令牌通过到期时间、颁发者和数字签名等对象来指示令牌的有效性。签名（`.` 分隔的 JWT 的第三个也是最后一个分段）是令牌验证的关键组件。恶意用户会修改令牌，但如果您的应用程序检索公钥并比较签名，则该令牌将不匹配。任何通过 OIDC 身份验证进行处理 JWTs 的应用程序都必须在每次登录时执行此验证操作。

在本页上，我们提出了一些一般和具体的验证建议 JWTs。在应用程序开发过程中，开发人员需要使用各种编程语言和平台。由于 Amazon Cognito 实施的 OIDC 足够接近公共规范，因此选择开发人员环境中任何声誉良好的 JWT 库都可以满足您的验证要求。

这些步骤描述了验证用户池 JSON Web 令牌（JWT）的过程。

主题

- 先决条件
- 使用验证令牌 aws-jwt-verify
- 了解和检查令牌

先决条件

您的库、SDK 或软件框架可能已经处理了本节中的任务。 Amazon SDKs 为应用程序中的 Amazon Cognito 用户池令牌处理和管理提供工具。 Amazon Amplify 包括检索和刷新 Amazon Cognito 令牌的功能。

有关更多信息，请参阅以下页面。

- 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成
- 使用 Amazon Cognito 身份提供商的代码示例 Amazon SDKs
- Amplify Dev Center（Amplify 开发中心）中的 Advanced workflows（高级工作流）

许多库可用于解码和验证 JSON Web 令牌 (JWT)。如果您要手动处理用于服务器端 API 处理的令牌，或者您使用的是其他编程语言，则这些库可以为您提供帮助。请参阅用于处理 JWT 令牌库的 OpenID Foundation 列表。

使用验证令牌 aws-jwt-verify

在 Node.js 应用程序中， Amazon 建议 aws-jwt-verify 库，用于验证用户传递给您的应用程序的令牌中的参数。使用 `aws-jwt-verify`，您可以使用要为一个或多个用户群体验证的声明值填充 `CognitoJwtVerifier`。它可以检查的一些值包括以下内容。

- 该访问令牌或 ID 令牌的格式不正确或已过期，但具有有效的签名。
- 这些访问令牌来自正确的用户群体和应用程序客户端。
- 该访问令牌声明包含正确的 OAuth 2.0 范围。
- 对访问令牌和 ID 令牌进行签名的密钥与用户群体的 JWKS URI 中的签名密钥 `kid` 匹配。

  JWKS URI 包含有关对用户令牌进行签名的私有密钥的公有信息。您可以在以下位置找到用户群体的 JWKS URI：`https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json`。

有关您可以在 Node.js 应用程序或 Amazon Lambda 授权方中使用的更多信息和示例代码，请参阅 aws-jwt-verify上 GitHub。

了解和检查令牌

在将令牌检查与应用程序集成之前，请考虑一下 Amazon Cognito 是如何组装的。 JWTs从用户群体中检索示例令牌。解码并详细检查它们，以了解它们的特征，并确定要验证的内容和时间。例如，您可能希望检查一个场景中的组成员资格，而在另一个场景中，您可能想要检查范围。

以下各节描述了在准备应用程序时手动检查 Amazon Cognito JWTs 的过程。

确认 JWT 的结构

JSON Web 令牌 (JWT) 包括三个部分，各部分之间有一个 `.`（圆点）分隔符。

标题

Amazon Cognito 用来对令牌进行签名的密钥 ID `kid` 和 RSA 算法 `alg`。Amazon Cognito 使用 `alg` (`RS256`) 对令牌进行签名。`kid` 是对您的用户池持有的 2048 位 RSA 私有签名密钥的截断引用。

有效负载

令牌声明。在 ID 令牌中，声明包括用户属性和有关用户群体 `iss` 和应用程序客户端 `aud` 的信息。在访问令牌中，有效负载包括范围、组成员资格、用户群体身份 (`iss`) 和应用程序客户端 (`client_id`)。

签名

签名不能像标头和有效载荷那样解码 base64url。它是从签名密钥和参数派生的 RSA256 标识符，您可以在 JWKS URI 中观察到这些标识符。

标头和有效负载是 base64url 编码的 JSON。您可以通过解码为起始字符 `eyJ` 的开头字符 `{` 来识别它们。如果您的用户向您的应用程序提供了 base64url 编码的 JWT，但该格式不正确，则它不是有效的亚马逊 Cognito 令牌`[JSON Header].[JSON Payload].[Signature]`，您可以将其丢弃。

以下示例应用程序使用验证用户池令牌。`aws-jwt-verify`

```
// cognito-verify.js
// Usage example: node cognito-verify.js eyJra789ghiEXAMPLE

const { CognitoJwtVerifier } = require('aws-jwt-verify');

// Replace with your Amazon Cognito user pool ID
const userPoolId = 'us-west-2_EXAMPLE';

async function verifyJWT(token) {
  try {
    const verifier = CognitoJwtVerifier.create({
      userPoolId,
      tokenUse: 'access', // or 'id' for ID tokens
      clientId: '1example23456789, // Optional, only if you need to verify the token
 audience
    });

    const payload = await verifier.verify(token);
    console.log('Decoded JWT:', payload);
```

```
  } catch (err) {
    console.error('Error verifying JWT:', err);
  }
}

// Example usage
if (process.argv.length < 3) {
  console.error('Please provide a JWT token as an argument.');
  process.exit(1);
}

const MyToken = process.argv[2];
verifyJWT(MyToken);
```

验证 JWT

JWT 签名是标头和负载的哈希组合。Amazon Cognito 为每个用户池生成两对 RSA 加密密钥。一个私有密钥对访问令牌进行签名，另一个私有密钥对 ID 令牌进行签名。

验证 JWT 令牌的签名

1. 解码 ID 令牌。

   OpenID Foundation 还维护用于处理 JWT 令牌的库列表。

   您也可以使用 Amazon Lambda 解码用户池 JWTs。有关更多信息，请参阅使用解码和验证 Amazon Cognito J WT 令牌。 Amazon Lambda

2. 将本地密钥 ID (`kid`) 与公有 `kid` 进行比较。

   a. 下载并存储适用于用户池的对应的公有 JSON Web Key（JWK）。它可作为 JSON Web Key Set (JWKS) 的一部分提供。您可以通过为您的环境构建以下 `jwks_uri` URL 来找到它：

      ```
      https://cognito-idp.<Region>.amazonaws.com/<userPoolId>/.well-known/jwks.json
      ```

      有关更多 JWK 和 JWK 集的更多信息，请参阅 JSON Web Key (JWK)。

      > **Note**
      >
      > Amazon Cognito 可能会轮换用户群体中的签名密钥。最佳做法是使用 `kid` 作为缓存密钥在应用程序中缓存公有密钥，并定期刷新缓存。将您的应用程序收到的令牌中的 `kid` 与缓存进行比较。

如果您收到的令牌的颁发者是正确的，但 kid 不同，则 Amazon Cognito 可能已经轮换了签名密钥。从您的用户群体 jwks_uri 端点刷新缓存。

这是个 jwks.json 文件示例：

```
{
 "keys": [{
  "kid": "1234example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "1234567890",
  "use": "sig"
 }, {
  "kid": "5678example=",
  "alg": "RS256",
  "kty": "RSA",
  "e": "AQAB",
  "n": "987654321",
  "use": "sig"
 }]
}
```

密钥 ID（**kid**）

kid 是一个提示，指示哪个密钥用于保护令牌的 JSON Web 签名（JWS）。

算法（**alg**）

alg 标头参数表示用于保护 ID 令牌的加密算法。用户池使用 RS256加密算法，即带有 SHA-256 的 RSA 签名。有关 RSA 的更多信息，请参阅 RSA 加密。

密钥类型（**kty**）

kty 参数标识与密钥结合使用的加密算法系列，例如，在本示例中为“RSA”。

RSA 指数（**e**）

e 参数包含 RSA 公有密钥的指数值。它以 base64URL UInt 编码的值表示。

RSA 模数（**n**）

n 参数包含 RSA 公有密钥的模数值。它以 base64URL UInt 编码的值表示。

使用（**use**）

use 参数描述了公有密钥的预期用途。在本示例中，use 值 sig 表示签名。

b. 搜索与您 JWT 的 kid 相匹配的 kid 的公有 JSON Web 密钥。

验证声明

验证 JWT 声明

1. 通过以下方法之一，验证令牌是否未过期。

   a. 对令牌解码并将 exp 声明与当前时间进行比较。

   b. 如果您的访问令牌包含aws.cognito.signin.user.admin索赔，请向类似的 API 发送请求GetUser。如果令牌已过期，您使用访问令牌进行授权的 API 请求会返回错误。

   c. 在针对userInfo 端点的请求中提供您的访问令牌。如果您的令牌已过期，则请求会返回错误。

2. ID 令牌中的 aud 声明和访问令牌中的 client_id 声明应与在 Amazon Cognito 用户池中创建的应用程序客户端 ID 匹配。

3. 发布者 (iss) 声明应与您的用户池匹配。例如，在 us-east-1 区域中创建的用户池将有下列 iss 值：

   https://cognito-idp.us-east-1.amazonaws.com/*<userpoolID>*.

4. 检查 token_use 声明。

   - 如果您在 Web API 操作中只接受访问令牌，则其值必须为 access。
   - 如果您只使用 ID 令牌，则其值必须为 id。
   - 如果您同时使用 ID 令牌和访问令牌，则 token_use 声明必须为 id 或 access。

您现在可以信任该令牌内的声明。

管理用户池令牌到期和缓存

每次您想要获取新的 JSON Web 令牌（JWT）时，应用程序都必须成功完成以下请求之一。

- 从令牌端点请求客户端凭证或授权代码授予。
- 从您的托管登录页面申请隐式授权。
- 在 Amazon Cognito API 请求中对本地用户进行身份验证，例如。InitiateAuth

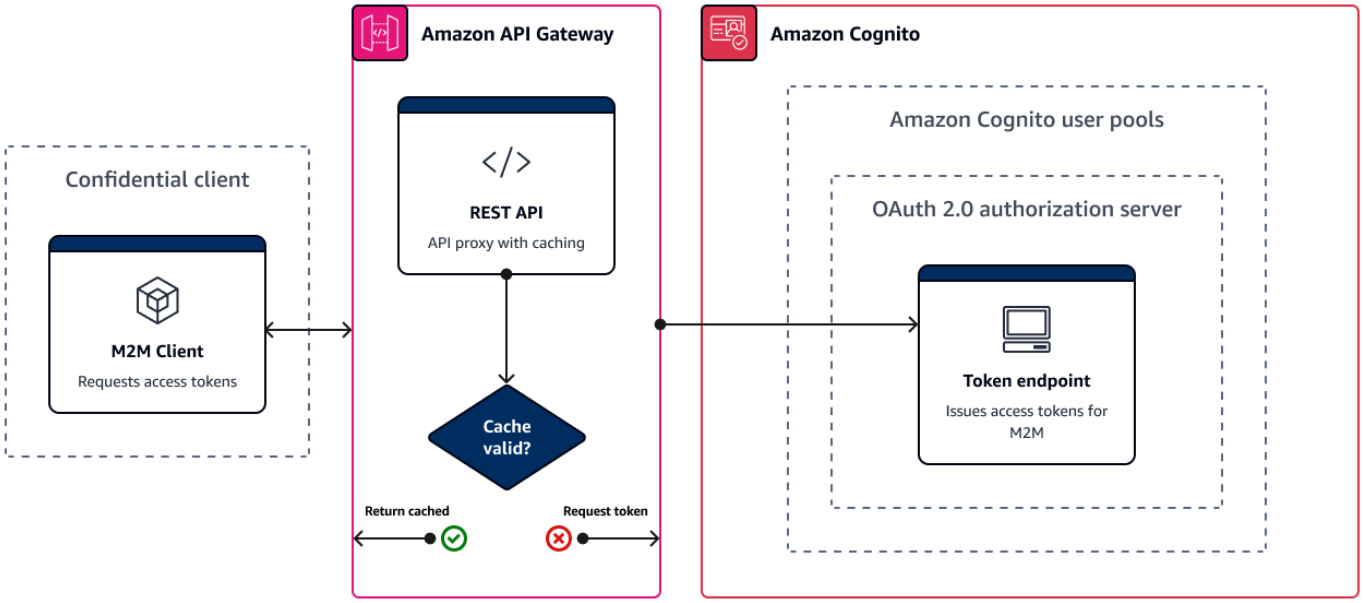
您可以将用户群体配置为将令牌设置为在数分钟、数小时或数天后过期。要确保应用程序的性能和可用性，请使用 Amazon Cognito 令牌至大约 75% 的令牌生命周期，并且仅在那时获取新令牌。您为应用程序构建的缓存解决方案可保持令牌可用，并可防止 Amazon Cognito 在您的请求速率过高时拒绝请求。客户端应用程序必须将令牌存储在内存缓存中。服务器端应用程序可以添加加密的缓存机制来存储令牌。

当您的用户池生成大量用户或 machine-to-machine活动时，您可能会遇到 Amazon Cognito 对您可以发出的令牌请求数量设定的限制。要减少您向 Amazon Cognito 端点发出的请求数量，可以安全地存储和重复使用身份验证数据，也可以实施指数回退和重试。

身份验证数据来自两类端点。Amazon Cognito OAuth2.0 终端节点包括令牌终端节点，该终端节点为客户端凭证和托管登录授权码请求提供服务。服务端点应答用户群体 API 请求，例如 `InitiateAuth` 和 `RespondToAuthChallenge`。每种类型的请求都有自己的限制。有关 限制的更多信息，请参阅 Amazon Cognito 中的限额。

使用 Amazon API Gateway 缓存 machine-to-machine访问令牌

借助 API Gateway 令牌缓存，您的应用程序可以扩展以响应大于 Amazon Cognito OAuth 终端节点默认请求速率配额的事件。



您可以缓存访问令牌，以便您的应用程序仅在缓存的令牌过期时才请求新的访问令牌。否则，您的缓存端点会从缓存中返回一个令牌。这会阻止对 Amazon Cognito API 端点的其他调用。当您使用 Amazon API Gateway 作为 令牌端点的代理时，您的 API 会响应大多数原本会计入您的请求配额的请求，从而避免由于速率限制而导致的请求失败。

以下基于 API Gateway 的解决方案提供了低延迟、低代码/无代码的令牌缓存实施。API Gateway APIs 在传输过程中进行加密，也可以选择静态加密。API Gateway 缓存非常适合 OAuth 2.0 客户端凭证授予，这是一种通常是大批量的授予类型，用于生成用于授权 machine-to-machine和微服务会话的访问令牌。在诸如流量激增导致您的微服务水平扩展的情况下，您最终可能会有许多系统使用相同的客户端凭据，但其数量超过用户池或应用程序客户端的 Amazon 请求速率限制。要保持应用程序可用性和低延迟，缓存解决方案是此类情况下的最佳实践。

在此解决方案中，您可以在 API 中定义一个缓存，以便为要在应用中请求的 OAuth 范围和应用程序客户端的每种组合存储单独的访问令牌。当您的应用程序发出与缓存键匹配的请求时，您的 API 会使用 Amazon Cognito 向与缓存键匹配的第一个请求颁发的访问令牌进行响应。当您的缓存键持续时间到期时，API 会将请求转发到令牌端点并缓存新的访问令牌。

> **Note**
>
> 您的缓存键持续时间必须短于应用程序客户端的访问令牌持续时间。

缓存密钥是您在请求正文`scope`参数中请求的 OAuth 作用域和请求`Authorization`标头的组合。`Authorization` 标头包含您的应用程序客户端 ID 和客户端密钥。您无需在应用程序中实施其他逻辑即可实施此解决方案。您只必须更新配置以更改用户群体令牌端点的路径。

您也可以使用 ElastiCache (Redis OSS) 实现令牌缓存。要使用 Amazon Identity and Access Management (IAM) 策略进行精细控制，请考虑 Amazon DynamoDB 缓存。

> **Note**
>
> 在 API Gateway 中缓存需要支付额外费用。有关更多详细信息，请参阅定价。

使用 API Gateway 设置缓存代理

1. 打开 API Gateway 控制台，然后创建一个 REST API。
2. 在 Resources（资源）中，创建一个 POST 方法。
   a. 选择 HTTP Integration type（集成类型）。
   b. 选择 Use HTTP proxy integration（使用 HTTP 代理集成）。
   c. 输入 Endpoint URL（端点 URL）`https://<your user pool domain>/oauth2/token`。

3. 在 Resources（资源）中，配置缓存键。
   a. 编辑 POST 方法的 Method request（方法请求）。
   b. 将您的 `scope` 参数和 `Authorization` 标头设置为缓存键。
      i. 在 URL 查询字符串参数中添加查询字符串。输入查询字符串 “名称”，然后选择 “必填项” `scope` 和 “缓存”。
      ii. 在 HTTP 请求标头中添加标头。输入请求标头的名称，`Authorization`然后选择必填和缓存。
4. 在 Stages（阶段）中，配置缓存。
   a. 选择要修改的舞台，然后从 “舞台细节” 中选择 “编辑”。
   b. 在 “其他设置” 下的 “缓存设置” 下，打开 “配置 API 缓存” 选项。
   c. 选择 Cache capacity（缓存容量）。较高的缓存容量可以提高性能，但会带来额外的成本。
   d. 清除需要授权复选框。选择继续。
   e. API Gateway 仅将缓存策略应用于阶段级别的 GET 方法。您必须对您的 POST 方法应用缓存策略替代。

      展开您配置的阶段并选择`POST`方法。要为该方法创建缓存设置，请选择创建覆盖。
   f. 激活 “启用方法缓存” 选项。
   g. 输入 3600 秒的缓存 time-to-live (TTL)。选择保存。
5. 在 Stages（阶段）中，注意 Invoke URL（调用 URL）。
6. 更新您的应用程序，以将令牌请求发布到 API 的 Invoke URL（调用 URL）而不是用户群体的 `/oauth2/token` 端点。

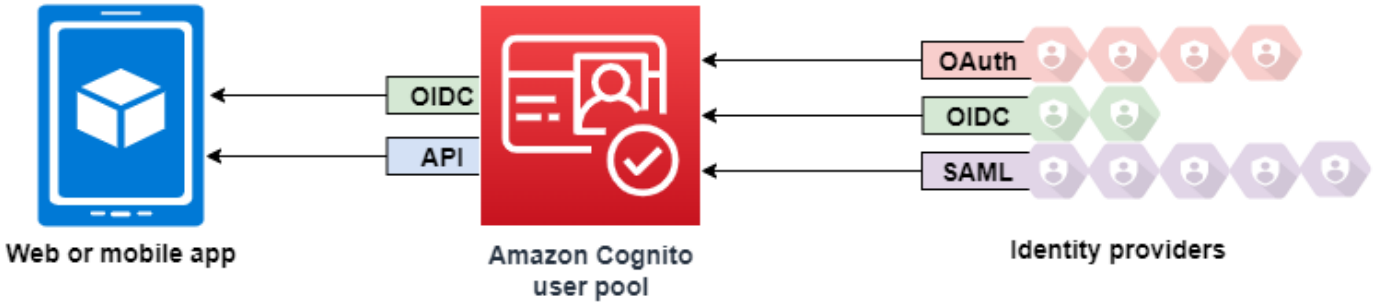
在成功登录后访问资源

您的应用程序用户可以通过用户池直接登录，也可以通过第三方身份提供者（IdP）进行联合身份验证。用户池管理处理通过Facebook、谷歌、亚马逊和苹果进行社交登录以及从OpenID Connect（OIDC）和SAML返回的代币的开销。 IdPs有关更多信息，请参阅 了解用户池 JSON 网络令牌 (JWTs)。

成功进行身份验证后，您的应用程序将收到来自 Amazon Cognito 的用户池令牌。您可以使用用户池令牌来完成以下操作：

- 在 Amazon DynamoDB 和 Amazon S3 Amazon Web Services 服务 等中检索授权应用程序资源请求的 Amazon 证书。
- 提供临时、可撤销的身份验证证明。
- 将身份数据填入到应用程序中的用户配置文件中。
- 授权对已登录用户在用户池目录中的配置文件进行更改。
- 使用访问令牌授权对用户信息的请求。
- 使用访问令牌授权对位于受访问保护的外部 APIs 数据之后的请求。
- 使用 Amazon Verified Permissions 授权对存储在客户端或服务器上的应用程序资产的访问。

有关更多信息，请参阅身份验证会话示例 和了解用户池 JSON 网络令牌 (JWTs)。



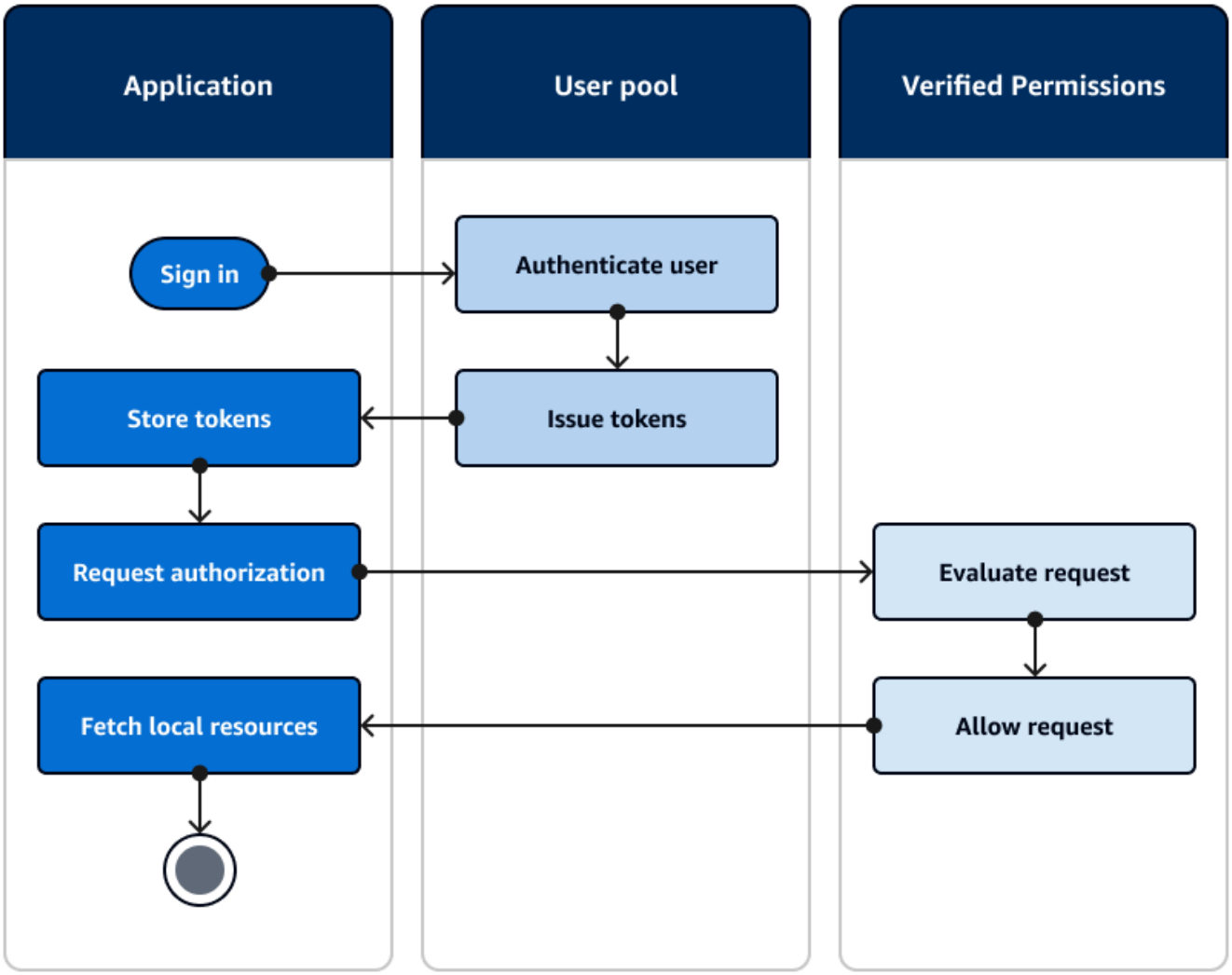
主题

- 使用 Amazon Verified Permissions 授权对客户端或服务器资源的访问。
- 在登录后使用 API Gateway 访问资源
- 在登录后使用身份池访问 Amazon Web Services 服务

使用 Amazon Verified Permissions 授权对客户端或服务器资源的访问。

您的应用程序可以将已登录用户的令牌传递给 Amazon Verified Permissions。Verified Permissions 是一项可扩展、精细的权限管理和授权服务，适用于您构建的应用程序。Amazon Cognito 用户池可以作为 Verified Permissions 策略存储的身份源。Verified Permissions 会根据用户池令牌中的主体及其属性对请求的操作和资源（例如 `premium_badge.png` 的 `GetPhoto`）作出授权决策。

下图显示了您的应用程序如何在授权请求中将用户的令牌传递给 Verified Permissions。



Amazon Verified Permissions 入门

将用户池与 Verified Permissions 集成后，您可以集中管理所有 Amazon Cognito 应用程序的精细授权。这样就不需要在所有应用程序中重复编写和复制精细安全逻辑。有关使用 Verified Permissions 进行授权的更多信息，请参阅使用 Amazon Verified Permissions 进行授权。

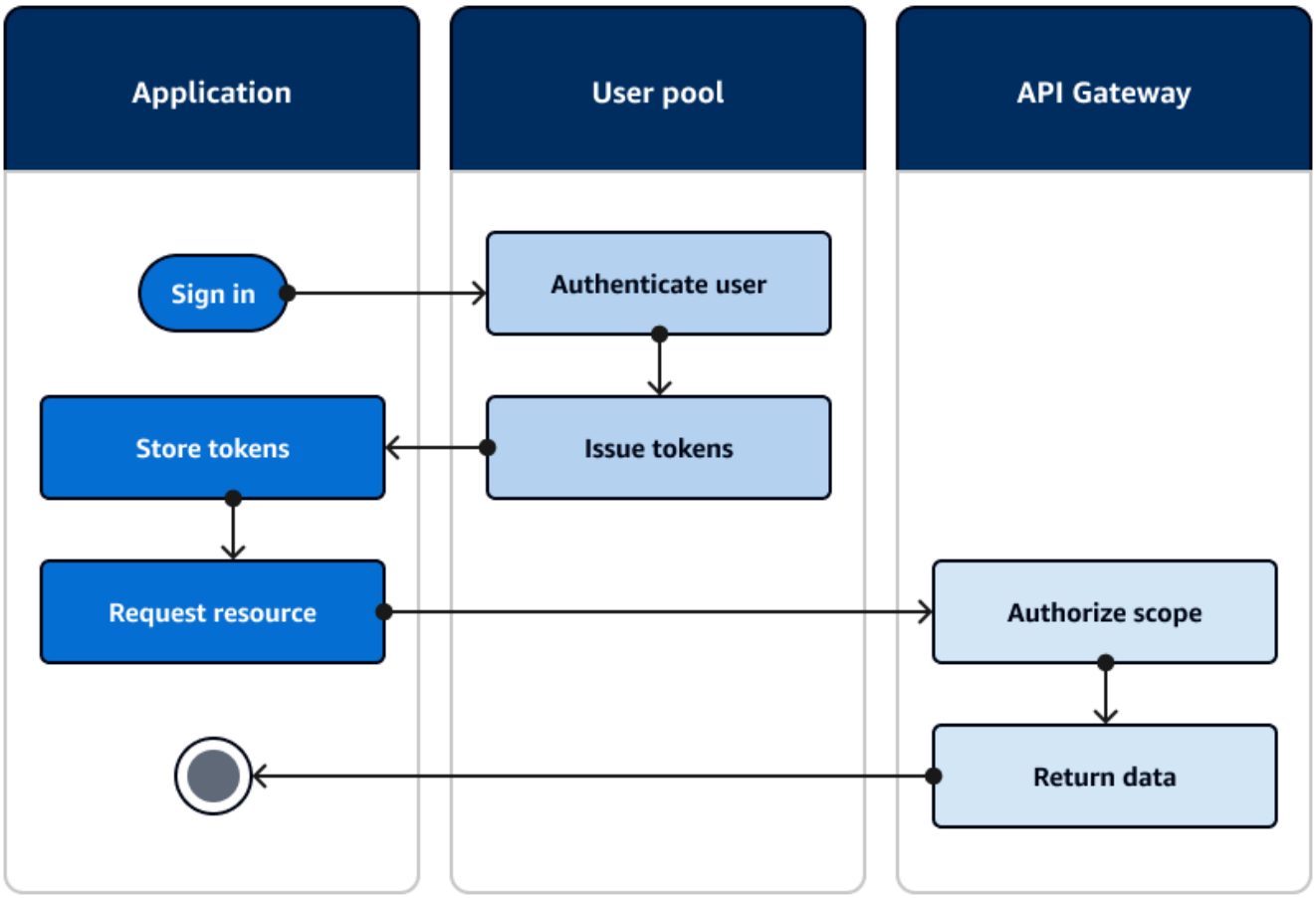
已验证的权限授权请求需要 Amazon 凭证。您可以实施以下一些技术来安全地将凭证应用于授权请求。

- 操作可以在服务器后端存储密钥的 Web 应用程序。
- 获取经过身份验证的身份池凭证。
- 通过 access-token-authorized API 代理用户请求，并将 Amazon 凭据附加到请求中。

在登录后使用 API Gateway 访问资源

Amazon Cognito 用户池令牌的常见用途是授权向 API Gateway REST API 发出的请求。访问令牌中的 OAuth 2.0 作用域可以授权方法和路径，HTTP GET例如/app_assets。ID 令牌可以用作 API 的通用身份验证，也可以将用户属性传递给后端服务。API Gateway 还有其他自定义授权选项，例如适用于 HTTP 的 JWT 授权方和 APIs Lambda 授权者，它们可以应用更精细的逻辑。

下图说明了访问令牌中具有 OAuth 2.0 范围的 REST API 的应用程序。



您的应用程序必须从经过身份验证的会话中收集令牌，并将其作为持有者令牌添加到请求中的 Authorization 标头。配置您为用于评估令牌内容的 API、路径和方法配置的授权方。仅当请求符合您为授权方设置的条件时，API Gateway 才会返回数据。

API Gateway API 可以通过以下一些方法来批准来自应用程序的访问：

- 访问令牌有效，未过期，并且包含正确的 OAuth 2.0 范围。适用于 REST API 的 Amazon Cognito 用户池授权方是一种常见的实现，进入门槛很低。您还可以评估向此类授权方发出的请求的正文、查询字符串和标头。
- ID 令牌有效且未过期。当您将 ID 令牌传递给 Amazon Cognito 授权方时，您可以在应用程序服务器上对 ID 令牌内容进行额外验证。
- 访问令牌或 ID 令牌中的组、声明、属性或角色符合您在 Lambda 函数中定义的要求。Lambda 授权方解析请求标头中的令牌，并对其进行评估以作出授权决策。您可以在函数中构造自定义逻辑，也可以向 Amazon Verified Permissions 发出 API 请求。

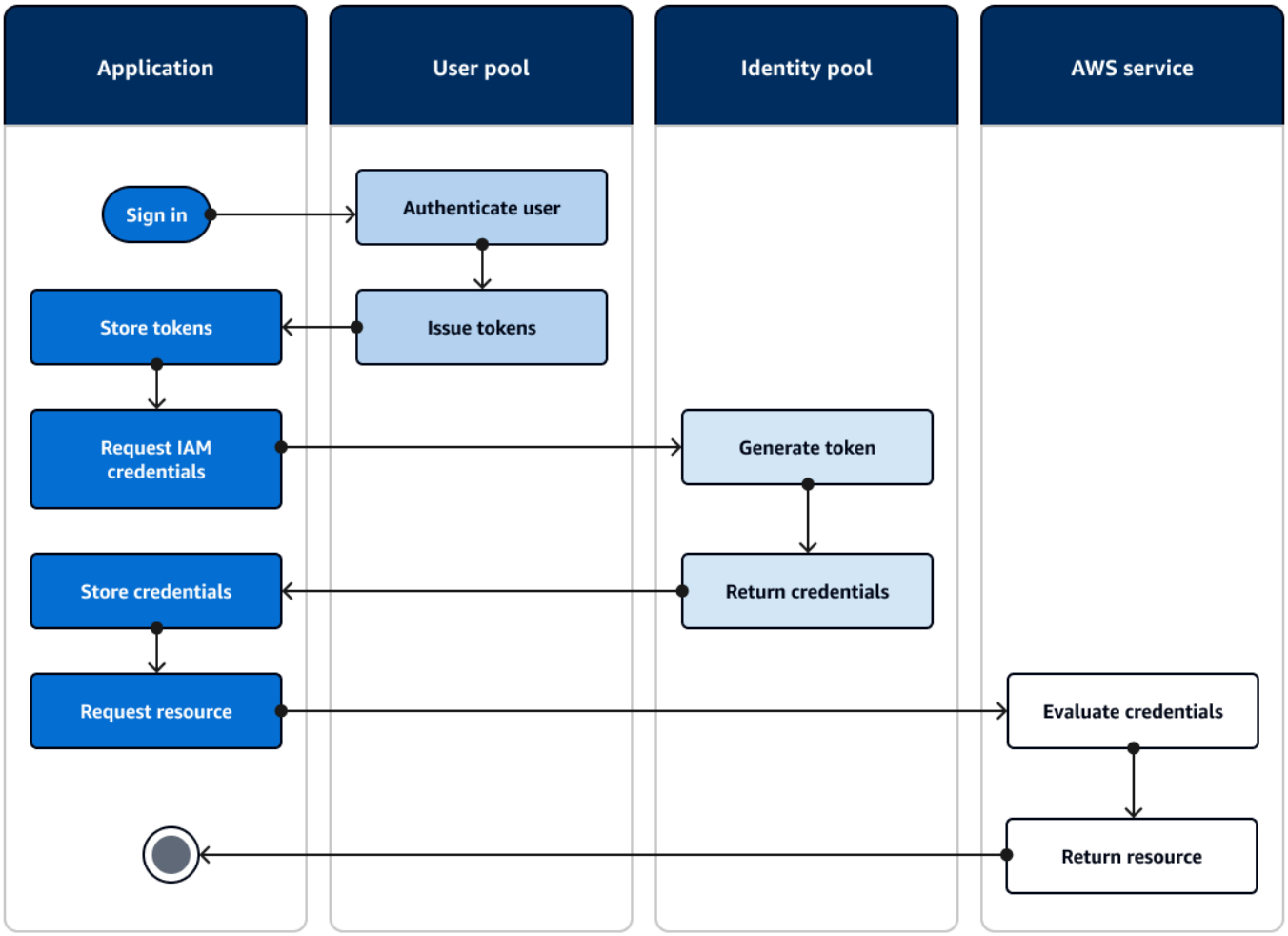
您还可以使用来自用户池的令牌来授权向 Amazon AppSync GraphQL API 发出的请求。

在登录后使用身份池访问 Amazon Web Services 服务

用户使用用户池登录后，他们可以使用身份池颁发的临时 API 凭证进行访问 Amazon Web Services 服务 。

您的 Web 应用程序或移动应用程序接收来自用户池的令牌。当您将用户池配置为身份池的身份提供者时，身份池会用令牌交换临时 Amazon 证书。这些证书的范围可以限于 IAM 角色及其策略，这些策略允许用户访问有限的 Amazon 资源。有关更多信息，请参阅 身份池身份验证流程。

下图显示了应用程序如何使用用户池登录、检索身份池凭证以及如何从 Amazon Web Services 服务请求资产。



您可以使用身份池凭证来完成以下操作：

- 使用用户自己的凭证向 Amazon Verified Permissions 提出精细的授权请求。
- 连接到 Amazon API Gateway REST API 或授权与 IAM 连接的 Amazon AppSync GraphQL API。
- 连接到使用 IAM 授权连接的数据库后端，例如 Amazon DynamoDB 或 Amazon RDS。
- 从 Amazon S3 存储桶检索应用程序资产。
- 使用 Amazon WorkSpaces 虚拟桌面启动会话。

身份池不仅在经过用户池认证的会话中运行。它们还直接接受来自第三方身份提供者的身份验证，并且可为未经身份验证的访客用户生成凭证。

有关使用身份池和用户池组来控制对 Amazon 资源的访问的更多信息，请参阅向用户池添加组和使用基于角色的访问控制。另外，有关身份池和的更多信息 Amazon Identity and Access Management，请参阅身份池身份验证流程。

使用设置用户池 Amazon Web Services Management Console

创建 Amazon Cognito 用户池并记下每个客户端应用程序的用户池 ID 和应用程序客户端 ID。有关创建用户池的更多信息，请参阅用户池入门。

使用设置身份池 Amazon Web Services Management Console

以下过程介绍如何使用将 Amazon Web Services Management Console 身份池与一个或多个用户池和客户端应用程序集成。

添加 Amazon Cognito 用户群体身份提供者 (IdP)

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Amazon Cognito 用户群体。
5. 输入用户群体 ID 和应用程序客户端 ID。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。

   a. 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来将声明与规则进行比较的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 如果您选择选择令牌中有 preferred_role 声明的角色，Amazon Cognito 会在您的用户的 `cognito:preferred_role` 声明中为该角色发放凭证。如果不存在首选角色声明，Amazon Cognito 将根据您的角色解析发放凭证。

   b. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   - 如果不应用主体标签，请选择非活动。
   - 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   - 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

将用户池与身份池集成

对您的应用程序用户进行身份验证后，将用户的身份令牌添加到凭证提供程序中的登录映射中。提供商名称取决于 Amazon Cognito 用户池 ID。结构如下所示：

```
cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>
```

您可以`<region>`从用户池 ID 中推导出值。例如，如果用户池 ID 为`us-east-1_EXAMPLE1`，则`<region>`为`us-east-1`。如果用户池 ID 为`us-west-2_EXAMPLE2`，则`<region>`为`us-west-2`。

JavaScript

```
var cognitoUser = userPool.getCurrentUser();

if (cognitoUser != null) {
 cognitoUser.getSession(function(err, result) {
  if (result) {
   console.log('You are now logged in.');

   // Add the User's Id Token to the Cognito credentials login map.
   AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'YOUR_IDENTITY_POOL_ID',
    Logins: {
     'cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>':
 result.getIdToken().getJwtToken()
    }
   });
  }
 });
```

```
}
```

Android

```
cognitoUser.getSessionInBackground(new AuthenticationHandler() {
 @Override
 public void onSuccess(CognitoUserSession session) {
  String idToken = session.getIdToken().getJWTToken();

  Map<String, String> logins = new HashMap<String, String>();
  logins.put("cognito-idp.<region>.amazonaws.com/<YOUR_USER_POOL_ID>",
 session.getIdToken().getJWTToken());
  credentialsProvider.setLogins(logins);
 }

});
```

iOS - objective-C

```
AWSServiceConfiguration *serviceConfiguration = [[AWSServiceConfiguration alloc]
 initWithRegion:AWSRegionUSEast1 credentialsProvider:nil];
AWSCognitoIdentityUserPoolConfiguration *userPoolConfiguration =
 [[AWSCognitoIdentityUserPoolConfiguration alloc] initWithClientId:@"YOUR_CLIENT_ID"
  clientSecret:@"YOUR_CLIENT_SECRET" poolId:@"YOUR_USER_POOL_ID"];
[AWSCognitoIdentityUserPool
 registerCognitoIdentityUserPoolWithConfiguration:serviceConfiguration
 userPoolConfiguration:userPoolConfiguration forKey:@"UserPool"];
AWSCognitoIdentityUserPool *pool = [AWSCognitoIdentityUserPool
 CognitoIdentityUserPoolForKey:@"UserPool"];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc] initWithRegionType:AWSRegionUSEast1 identityPoolId:@"YOUR_IDENTITY_POOL_ID"
 identityProviderManager:pool];
```

iOS - swift

```
let serviceConfiguration = AWSServiceConfiguration(region: .USEast1,
 credentialsProvider: nil)
let userPoolConfiguration = AWSCognitoIdentityUserPoolConfiguration(clientId:
 "YOUR_CLIENT_ID", clientSecret: "YOUR_CLIENT_SECRET", poolId: "YOUR_USER_POOL_ID")
AWSCognitoIdentityUserPool.registerCognitoIdentityUserPoolWithConfiguration(serviceConfigura
 userPoolConfiguration: userPoolConfiguration, forKey: "UserPool")
let pool = AWSCognitoIdentityUserPool(forKey: "UserPool")
```

```
let credentialsProvider = AWSCognitoCredentialsProvider(regionType: .USEast1,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", identityProviderManager:pool)
```

配置用户池功能

在前面的章节中，您可能已经在 Amazon Cognito 控制台的指导下配置了一些功能。本节中的页面更深入地探讨了用户池的一些核心功能的详细配置要求。这里有关于你在应用程序客户端、电子邮件和短信配置、记住用户设备等方面的选项的重要参考信息。

主题

- 更新用户池和应用程序客户端配置
- 特定于应用程序的应用程序客户端设置
- 使用用户群体中的用户设备
- 作用域、M2M 以及 APIs 带资源服务器的场景
- 使用 Amazon Pinpoint 进行用户池分析
- Amazon Cognito 用户池的电子邮件设置
- Amazon Cognito 用户池的短信设置

更新用户池和应用程序客户端配置

当您想要更改用户池或应用程序客户端中的设置时，只需点击几下即可在 Amazon Cognito 控制台中应用更新。浏览用户池设置中基于特征的选项卡，并按照本指南其他部分的说明更新字段。

许多组织通过编程方式管理其资源 Amazon CloudFormation、基于 Amazon SDKs 或 CDK 构建的应用程序以及其他自动化软件。如果这是您的资源管理模型，则在资源中逐步实施变更时必须格外小心。

API 操作 UpdateUserPool UpdateUserPoolClient并更新现有用户池或应用程序客户端。每个操作在 API 参考中都有一个警告：如果您没有为属性提供值，Amazon Cognito 会将其设置为默认值。当您提交仅包含一个参数的更新请求时，Amazon Cognito 会将该参数设置为您选择的值，并将所有其他参数设置为默认值。这样可以重置配置，包括您的属性架构、Lambda 触发器以及电子邮件和短信消息配置。

此外，在创建用户池或应用程序客户端后，有些设置会锁定，除非创建新资源，否则无法更改这些设置。

主题

- 无法更改的设置
- 短信配置
- 使用 Amazon SDK 或 REST API 更新用户池 Amazon CDK

无法更改的设置

创建用户群体后，您无法更改一些设置。如果您想要更改以下设置，您必须创建新的用户群体或应用程序客户端。

> **Note**
>
> 以前，您无法更改用户池的名称。这种情况已经改变。现在，您可以为用户池分配新的友好名称。

用户池 ID

API 参数名称：ID/ UserPoolId

例如`us-east-1_EXAMPLE`，用户池 ID 由 Amazon Cognito 自动生成，无法更改。

Amazon Cognito 用户群体登录选项

API 参数名称：AliasAttributes和 UsernameAttributes

用户登录时可以作为用户名传递的属性。创建用户群体时，您可以选择允许使用用户名、电子邮件地址、电话号码或首选用户名进行登录。要更改用户群体登录选项，请创建新的用户群体。

Make user name case sensitive（使用户名区分大小写）

API 参数名称：UsernameConfiguration

当您创建的用户名与其他用户名（字母大小写除外）匹配时，Amazon Cognito 可以将其视为同一用户或唯一用户。有关更多信息，请参阅 用户池区分大小写。要更改区分大小写，请创建新的用户群体。

客户端密钥

API 参数名称：GenerateSecret

创建应用程序客户端时，可以生成客户端密钥，以便只有受信任的来源才能向用户群体发出请求。有关更多信息，请参阅 特定于应用程序的应用程序客户端设置。要更改客户端密钥，请在同一用户群体中创建新的应用程序客户端。

必需的属性

API 参数名称：Schema

当用户注册时或当您创建属性时，用户必须为这些属性提供值。有关更多信息，请参阅 使用用户属性。要更改必需的属性，请创建新的用户群体。

自定义属性（删除）

API 参数名称：Schema

具有自定义名称的属性。您可以更改用户自定义属性的值，但不能从用户群体中删除自定义属性。有关更多信息，请参阅 使用用户属性。如果达到自定义属性的最大数量并且您想要修改列表，请创建新的用户群体。

短信配置

激活用户池中的短信后，您就无法停用它们。

- 如果您选择在创建用户池时配置短信，则在完成设置后无法停用短信。
- 您可以在自己创建的用户池中激活短信，但之后就无法停用短信。
- Amazon Cognito 可以使用短信进行用户账户邀请和恢复、属性验证和多重身份验证（MFA）。激活短信后，您可以随时为这些功能开启或关闭短信。
- 短信配置包括您委托给 Amazon Cognito 的 IAM 角色，以便使用 Amazon SNS 发送消息。您可以随时更改分配的角色。

使用 Amazon SDK 或 REST API 更新用户池 Amazon CDK

在 Amazon Cognito 控制台中，您可以更改用户池设置，一次更改一个参数。例如，要添加 Lambda 触发器，您可以选择添加 Lambda 触发器，然后选择函数和触发器类型。Amazon Cognito 用户池 API 按以下方式进行构造，用户池和应用程序客户端的更新操作需要用户池的完整参数集。但是，控制台会使用您的其他用户池设置透明地自动执行此更新操作。

有时你可能会发现，当更新与你要更改的设置无关时，你 Amazon Web Services 账户 可能会发现其他地方的更改可能会导致更新生成错误。例如，已删除的 Amazon SES 身份或 IAM 权限的 Amazon WAF更改。如果其中一个当前参数不再有效，则在修复该参数之前，您无法更新设置。遇到此类错误时，请检查错误响应并验证响应中提及的设置。

Amazon Cognito 用户池 REST API Amazon SDKs是用于自动化和编程配置 Amazon Cognito 资源的工具。Amazon Cloud Development Kit (Amazon CDK)使用这些工具的请求也必须像 Amazon Cognito 控制台一样，在请求正文中使用完整的资源配置来更新设置。概括来说，您必须执行以下步骤。

1. 从描述现有资源配置的操作中捕获输出。
2. 设置更改后修改输出。
3. 在更新资源的操作中发送修改后的配置。

以下过程使用 UpdateUserPoolAPI 操作更新您的配置。同样的方法适用于不同的输入字段 UpdateUserPoolClient。

> **Important**
>
> 如果您未为现有参数提供值，Amazon Cognito 将它们设置为默认值。例如，如果您有现有的 `LambdaConfig`，然后提交具有空 `LambdaConfig` 的 `UpdateUserPool`，则会删除为用户群体触发器分配的所有 Lambda 函数。当您想自动更改用户群体配置时，请相应地进行规划。

1. 使用捕获用户池的现有状态 DescribeUserPool。
2. 设置 `DescribeUserPool` 的输出的格式以与 `UpdateUserPool` 的请求参数匹配。从输出 JSON 中删除以下顶级字段及其子对象。
   - `Arn`
   - `CreationDate`
   - `CustomDomain`
     - 使用 UpdateUserPoolDomainAPI 操作更新此字段。
   - `Domain`
     - 使用 UpdateUserPoolDomainAPI 操作更新此字段。
   - `EmailConfigurationFailure`
   - `EstimatedNumberOfUsers`
   - `Id`
   - `LastModifiedDate`
   - `Name`
   - `SchemaAttributes`

- `SmsConfigurationFailure`
- `Status`

3. 确认生成的 JSON 与 `UpdateUserPool` 的请求参数匹配。
4. 修改任何您想要在生成的 JSON 中更改的参数。
5. 提交 `UpdateUserPool` API 请求，同时将您修改后的 JSON 作为请求输入。

您还可以在 Amazon CLI中，在 `update-user-pool` 的 `--cli-input-json` 参数中使用这一修改后的 `DescribeUserPool` 输出。

或者，运行以下 Amazon CLI 命令为接受的输入字段生成空值的 JSON。`update-user-pool`然后，您可以使用用户群体中的现有值填充这些字段。

```
aws cognito-idp update-user-pool --generate-cli-skeleton --output json
```

运行以下命令以为应用程序客户端生成相同的 JSON 对象。

```
aws cognito-idp update-user-pool-client --generate-cli-skeleton --output json
```

特定于应用程序的应用程序客户端设置

用户池应用程序客户端是用户池中的一项配置，它与一个通过 Amazon Cognito 进行身份验证的移动或 Web 应用程序进行交互。应用程序客户端可以调用经过授权和未经身份验证的 API 操作，并读取或修改用户的部分或全部属性。您的应用程序必须在操作中向应用程序客户端表明自己的身份，才能注册、登录和处理忘记的密码。这些 API 请求必须包括使用应用程序客户端 ID 进行自我识别以及使用可选客户端密钥进行授权的机制。您必须保护所有应用程序客户端 IDs 或密钥，以便只有经过授权的客户端应用程序才能调用这些未经身份验证的操作。此外，如果您将应用程序配置为使用 Amazon 凭证签署经过身份验证的 API 请求，则必须保护您的凭据免受用户检查。

您可以为一个用户池创建多个应用程序。应用程序客户端可能链接到应用程序的代码平台，也可能链接到用户池中的单独租户。例如，您可以为服务器端应用程序和其他 Android 应用程序创建一个应用程序。每个应用程序都有各自的应用程序客户端 ID。

您可以在应用程序客户端级别应用以下用户池特征的设置：

1. 分析
2. 托管登录 IdPs、授权类型 URLs、回调和自定义

3. 资源服务器和自定义范围
4. 威胁防护
5. 属性读取和写入权限
6. 令牌到期和撤销
7. 身份验证流程

应用程序客户端类型

在 Amazon Cognito 中创建应用程序客户端时，您可以根据标准 OAuth 客户端类型（公共客户端和机密客户端）预先填充选项。使用客户端密钥配置机密客户端 有关客户端类型的更多信息，请参阅 IETF RFC 6749 #2.1。

公有客户端

公有客户端在浏览器或移动设备上运行。由于它没有可信的服务器端资源，所以它没有客户端密钥。

机密客户端

机密客户端拥有可以信任的服务器端资源，使用客户端密钥进行未经身份验证的 API 操作。该应用程序可在后端服务器上作为守护进程或 Shell 脚本运行。

客户端密钥

客户端机密或客户端密码是一个固定字符串，您的应用程序必须在发送到应用程序客户端的所有 API 请求中使用该字符串。您的应用程序客户端必须有客户端密钥才能执行 `client_credentials` 授权。有关更多信息，请参阅 IETF RFC 6749 #2.3.1。

您在创建应用程序后无法更改密钥。如果要轮换密钥，您可以创建一个具备新私有密钥的新应用程序。您也可以删除应用程序，以便阻止使用该应用程序客户端 ID 的应用程序的访问。

> **Note**
>
> 当您为应用程序类型选择传统 Web 应用程序和 M 应用程序选项时，Amazon Cognito 控制台会创建带有客户端密钥的achine-to-machine应用程序客户端。选择以下选项之一来生成客户端密钥，或者使用编程方式创建客户端，CreateUserPoolClient并将其设置`GenerateSecret`为`true`。

您可以将机密客户端和客户端密钥用于公有应用程序。使用 Amazon CloudFront 代理添加SECRET_HASH在途中。有关更多信息，请参阅博客上的使用亚马逊 CloudFront 代理保护公共客户端 Amazon Cognito。 Amazon

JSON Web 令牌

亚马逊 Cognito 应用程序客户端可以发行以下类型的 JSON 网络令牌 (JWTs)。

身份（ID）令牌

一份可验证的声明，表明您的用户是从用户池进行的身份验证。OpenID Connect (OIDC) 在 2.0 定义的访问和刷新令牌标准中添加了 ID 令牌规范。 OAuth ID 令牌包含身份信息，例如用户属性，您的应用程序可以使用这些信息来创建用户个人资料和配置资源。请参阅了解身份（ID）令牌了解更多信息。

访问令牌

您的用户访问权限的可验证声明。访问令牌包含作用域、OIDC 和 OAuth 2.0 的功能。您的应用程序可以为后端资源提供范围，并证明您的用户池已授权用户或计算机访问来自 API 的数据或它们自己的用户数据。具有自定义范围的访问令牌（通常来自 M2M 客户端凭证授权）用于授权访问资源服务器。请参阅了解访问令牌了解更多信息。

刷新令牌

一种加密的初始身份验证声明，当您的用户令牌到期时，您的应用程序可以将其提供给您的用户池。刷新令牌请求会返回新的未到期访问令牌和 ID 令牌。请参阅刷新令牌了解更多信息。

您可以在 A mazon Cognito 控制台的用户池的应用程序客户端菜单中为每个应用程序客户端设置这些令牌的到期时间。

应用程序客户端术语

以下术语是 Amazon Cognito 控制台中应用程序客户端的可用属性。

允许的回调 URLs

回调 URL 指示在用户成功登录之后将被重新导向到哪里。选择至少一个回调 URL。回调 URL 必须：

- 是绝对 URI。
- 已预先向客户端注册。
- 不包含片段组件。

请参阅 OAuth 2.0-重定向端点。

Amazon Cognito 要求使用 HTTPS 而不是 HTTP，但 `http://localhost`（仅用于测试目的）除外。

还支持 URLs 诸如之`myapp://example`类的应用程序回调。

允许注销 URLs

注销 URL 指示在您的用户注销后会被重定向到哪里。

属性读取和写入权限

您的用户群可能有很多客户，每个客户都有自己的应用程序客户端，并且 IdPs。您可以将应用程序客户端配置为仅对与应用程序相关的用户属性具有读写权限。在 machine-to-machine（M2M）授权之类的情况下，您可以不授予对任何用户属性的访问权限。

属性读取和写入权限配置的注意事项

- 如果您创建应用程序客户端但不自定义属性读取和写入权限，Amazon Cognito 会向所有用户池属性授予读写权限。
- 您可以授予对不可变自定义属性的写入权限。创建或注册用户时，您的应用程序客户端可以将值写入不可变属性。此后，您将无法为用户的任何不可变自定义属性写入值。
- 应用程序客户端必须拥有对用户池中必要属性的写入权限。Amazon Cognito 控制台会自动将必要属性设置为可写属性。
- 您不能允许应用程序客户端对 `email_verified` 或 `phone_number_verified` 拥有写入权限。用户池管理员可以修改这些值。用户只能通过属性验证来更改这些属性的值。

身份验证流程

您的应用程序客户端允许的登录方法。您的应用程序可以支持使用用户名和密码进行身份验证、电子邮件和短信、密钥身份验证器 OTPs、使用 Lambda 触发器的自定义身份验证以及令牌刷新。作为最佳安全实践，请在定制应用程序中使用 SRP 身份验证进行用户名和密码身份验证。

自定义范围

自定义范围是您在资源服务器中为自己的资源服务器定义的范围。格式为*`resource-server-identifier`*/*`scope`*。请参阅 作用域、M2M 以及 APIs 带资源服务器的场景。

默认的重定向 URI

将用户身份验证请求中的`redirect_uri`参数替换为第三方 IdPs。使用CreateUserPoolClient或UpdateUserPoolClientAPI 请求的`DefaultRedirectURI`参数配置此应用程序客户端设置。此

URL 还必须是应用程序客户端的 `CallbackURLs` 的成员。在以下情况下，Amazon Cognito 会将经过身份验证的会话重定向到此 URL：

1. 您的应用程序客户端分配了一个身份提供商并 URLs定义了多个回调。如果身份验证请求不包括 `redirect_uri` 参数，则用户池会将对授权服务器的身份验证请求重定向到默认的重定向 URI。
2. 您的应用程序客户端分配了一个身份提供商并 URLs定义了一个回调。在这种情况下，无需定义默认的回调 URL。不包括 `redirect_uri` 参数的请求会重定向到一个可用的回调 URL。

身份提供者

您可以选择部分或全部用户池外部身份提供商 (IdPs) 来对用户进行身份验证。您的应用程序客户端还可以仅对用户池中的本地用户进行身份验证。向应用程序客户端添加 IdP 时，您可以生成指向 IdP 的授权链接并将其显示在您的托管登录登录页面上。您可以分配多个 IdPs，但必须至少分配一个。有关使用外部的更多信息 IdPs，请参阅用户池使用第三方身份提供商登录。

OpenID Connect 范围

选择以下一个或多个 `OAuth` 范围来指定可以为访问令牌请求的访问权限。

- `openid` 范围声明您要检索 ID 令牌和用户的唯一 ID。它还会请求全部或部分用户属性，具体取决于请求中的其他范围。除非您请求 `openid` 范围，否则 Amazon Cognito 不会返回 ID 令牌。`openid` 范围授权结构化 ID 令牌声明，例如过期时间和密钥 ID，并确定您在 userInfo 端点的响应中收到的用户属性。
  - 当 `openid` 是您请求的唯一范围时，Amazon Cognito 会使用当前应用程序客户端可以读取的所有用户属性填充 ID 令牌。对仅具有此范围的访问令牌的 `userInfo` 响应将返回所有用户属性。
  - 当您使用其他范围（例如 `phone`、`email` 或 `profile`）请求 `openid` 时，ID 令牌和 `userInfo` 返回用户的唯一 ID 以及由其他范围定义的属性。
- `phone` 范围授予对 `phone_number` 和 `phone_number_verified` 声明的访问权限。此范围只能通过 `openid` 范围来请求。
- `email` 范围授予对 `email` 和 `email_verified` 声明的访问权限。此范围只能通过 `openid` 范围来请求。
- 该`aws.cognito.signin.user.admin`范围允许访问需要访问令牌的 Amazon Cognito 用户池 API 操作，例如UpdateUserAttributes和。VerifyUserAttribute
- `profile` 范围授予对客户端可读取的所有用户属性的访问权限。此范围只能通过 `openid` 范围来请求。

有关范围的更多信息，请参阅标准 OIDC 范围列表。

OAuth 拨款类型

OAuth 授权是一种检索用户池令牌的身份验证方法。Amazon Cognito 支持以下类型的授权。要将这些 OAuth 授权集成到您的应用程序中，您必须向用户池中添加域名。

授予授权代码

授权码授权会生成一个代码，您的应用程序可以用它与令牌端点交换用户池令牌。当您交换授权码时，您的应用程序会收到 ID、访问权限和刷新令牌。与隐式授权一样，这种 OAuth 流程发生在用户的浏览器中。授权码授权是 Amazon Cognito 提供的最安全的授权，因为令牌在用户的会话中不可见。相反，您的应用程序会生成返回令牌的请求，并可以将其缓存在受保护存储空间中。有关更多信息，请参阅 IETF RFC 6749 #1.3.1 中的授权码。

> **Note**
>
> 作为公共客户端应用程序的最佳安全实践，仅激活授权码授予 OAuth 流程，并实施代码交换证明密钥 (PKCE) 以限制令牌交换。通过 PKCE，客户端只有在向令牌端点提供与原始身份验证请求中相同的机密时，才能交换授权码。有关 PKCE 的更多信息，请参阅 IETF RFC 7636。

隐式授予

隐式授权直接从对端点授权向用户的浏览器会话提供访问权限和 ID 令牌，但不返回刷新令牌。隐式授权消除了向令牌端点提出单独请求的要求，但与 PKCE 不兼容，也不会返回刷新令牌。该授权适用于无法完成授权码授权的测试场景和应用程序架构。有关更多信息，请参阅 IETF RFC 6749 #1.3.2 中的隐式授权。您可以在应用程序客户端中同时激活授权码授权和隐式授权，然后按需使用每个授权。

客户端凭证授权

客户端凭证授予用于 machine-to-machine (M2M) 通信。授权码和隐式授权向经过身份验证的人类用户发放令牌。客户端凭证授权非交互式系统对 API 的基于范围的授权。您的应用程序可以直接从令牌端点请求客户端凭证并接收访问令牌。有关更多信息，请参阅 IETF RFC 6749 #1.3.4 中的客户端凭证。您只能在具有客户端机密且不支持授权码或隐式授权的应用程序客户端中激活客户端凭证授权。

> **Note**
>
> 由于您没有以用户身份调用客户端凭证流程，因此该授权只能向访问令牌添加自定义范围。自定义范围就是您为自己的资源服务器定义的范围。默认范围（例如 `openid` 和 `profile`）不适用于非人类用户。
> 由于 ID 令牌是对用户属性的验证，因此它们与 M2M 通信无关，客户凭证授权也不会发放 ID 令牌。请参阅 作用域、M2M 以及 APIs 带资源服务器的场景。

客户凭证授予会增加您的 Amazon 账单费用。有关更多信息，请参阅 Amazon Cognito 定价。

创建应用程序客户端

Amazon Web Services Management Console

创建应用程序客户端（控制台）

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。这两个选项都会提示您使用特定于应用程序的设置来配置应用程序客户端。
4. 选择一种能反映您的应用程序架构的应用程序类型。
5. 使用友好的标识符@@ 命名您的应用程序。
6. 输入退货网址。
7. 选择创建应用程序客户端。创建应用程序客户端后，您可以更改高级选项。
8. Amazon Cognito 会将应用程序客户端信息返回给您。要访问应用程序的示例代码，请从 “快速设置指南” 选项卡中选择一个平台。

Amazon CLI

```
aws cognito-idp create-user-pool-client --user-pool-id MyUserPoolID --client-
name myApp
```

> **Note**
>
> 使用 JSON 格式进行回调和注销 URLs ，以防止 CLI 将其视为远程参数文件：

```
--callback-urls "["https://example.com"]"
--logout-urls "["https://example.com"]"
```

有关更多信息，请参阅 Amazon CLI 命令参考：create-user-pool-client

Amazon Cognito user pools API

生成 CreateUserPoolClientAPI 请求。必须为所有您不想设置为默认值的参数指定一个值。

更新用户池应用程序客户端（Amazon CLI 和 Amazon API）

在 Amazon CLI，输入以下命令：

```
aws cognito-idp update-user-pool-client --user-pool-id  "MyUserPoolID" --client-id
 "MyAppClientID" --allowed-o-auth-flows-user-pool-client --allowed-o-auth-flows "code"
 "implicit" --allowed-o-auth-scopes "openid" --callback-urls "["https://example.com"]"
 --supported-identity-providers "["MySAMLIdP", "LoginWithAmazon"]"
```

如果命令成功，则 Amazon CLI 返回确认信息：

```
{
    "UserPoolClient": {
        "ClientId": "MyClientID",
        "SupportedIdentityProviders": [
            "LoginWithAmazon",
            "MySAMLIdP"
        ],
        "CallbackURLs": [
            "https://example.com"
        ],
        "AllowedOAuthScopes": [
            "openid"
        ],
        "ClientName": "Example",
        "AllowedOAuthFlows": [
            "implicit",
            "code"
        ],
        "RefreshTokenValidity": 30,
        "AuthSessionValidity": 3,
```

```
        "CreationDate": 1524628110.29,
        "AllowedOAuthFlowsUserPoolClient": true,
        "UserPoolId": "MyUserPoolID",
        "LastModifiedDate": 1530055177.553
    }
}
```

有关更多信息，请参阅 Amazon CLI 命令参考:update-user-pool-client.

Amazon API: UpdateUserPoolClient

获取有关用户池应用程序客户端（Amazon CLI 和 Amazon API）的信息

```
aws cognito-idp describe-user-pool-client --user-pool-id MyUserPoolID --client-
id MyClientID
```

有关更多信息，请参阅 Amazon CLI 命令参考:describe-user-pool-client.

Amazon API: DescribeUserPoolClient

列出用户池（Amazon CLI 和 Amazon API）中的所有应用程序客户端信息

```
aws cognito-idp list-user-pool-clients --user-pool-id "MyUserPoolID" --max-results 3
```

有关更多信息，请参阅 Amazon CLI 命令参考:list-user-pool-clients.

Amazon API: ListUserPoolClients

删除用户池应用程序客户端（Amazon CLI 和 Amazon API）

```
aws cognito-idp delete-user-pool-client --user-pool-id "MyUserPoolID" --client-id
 "MyAppClientID"
```

有关更多信息，请参阅 Amazon CLI 命令参考：delete-user-pool-client

Amazon API: DeleteUserPoolClient

使用用户群体中的用户设备

当您使用 Amazon Cognito 用户池 API 登录本地用户池用户时，您可以将威胁防护中的用户活动日志与他们的每台设备相关联，如果您的用户使用的是可信设备，也可以允许他们跳过多重身份验证

(MFA)。对于任何尚未包含设备信息的登录，Amazon Cognito 都会在响应中包含设备密钥。设备密钥的格式为 *Region_UUID*。借助设备密钥、安全远程密码（SRP）库和允许设备身份验证的用户群体，您可以提示应用程序中的用户信任当前设备，而不再在登录时提示输入 MFA 代码。

主题

- 设置记忆设备
- 获取设备密钥
- 使用设备登录
- 查看、更新和忘记设备

设置记忆设备

借助 Amazon Cognito 用户群体，您可以将每个用户的设备与唯一的设备标识符（设备密钥）关联起来。当您在登录时出示设备密钥并执行设备身份验证时，可以为应用程序配置可信设备身份验证流程。在此流程中，您的应用程序可以让用户选择在不使用 MFA 的情况下登录，直到稍后再进行登录，具体取决于您的应用程序的安全要求或用户的偏好。在该时间段结束时，您的应用程序必须将设备状态更改为 “未记住”，并且用户必须使用 MFA 登录，直到他们确认要记住设备。例如，应用程序可能会提示用户信任某台设备 30 天、60 天或 90 天。您可以将此日期存储在自定义属性中，并在该日期更改其设备的记住状态。然后，必须重新提示用户提交 MFA 代码并将设备设置为在成功进行身份验证后再次记住设备。

1. 记住的设备只能在 MFA 处于活动状态的用户群体中覆盖 MFA。

当用户使用记住的设备登录时，您必须在其身份验证流程中执行额外的设备身份验证。有关更多信息，请参阅 使用设备登录。

将您的用户池配置为记住用户池登录菜单中的 “设备跟踪” 下的设备。通过 Amazon Cognito 控制台设置记忆设备功能时，有三种选项供您选择：Always (始终)、User Opt-In (用户选择加入) 和 No (否)。

请勿记住

用户群体不会提示用户在登录时记住设备。

始终记住

当应用程序确认用户的设备时，用户群体将始终记住该设备，并且不会在将来成功登录设备时返回 MFA 质询。

用户选择加入

当应用程序确认用户的设备后，用户群体不会自动抑制 MFA 质询。您必须提示用户选择是否要记住设备。

当您选择始终记住或用户选择加入时，每次用户从身份不明的设备登录时，Amazon Cognito 都会生成设备标识符密钥和机密。设备密钥是应用程序在用户执行设备身份验证时发送到用户群体的初始标识符。

对于每个已确认的用户设备，无论是自动记住还是选择加入，您都可以在每次用户登录时使用设备标识符密钥和机密对设备进行身份验证。

您还可以在CreateUserPool或 UpdateUserPoolAPI 请求中为用户池配置记忆设备设置。欲了解更多信息，请参阅该DeviceConfiguration属性。

Amazon Cognito 用户群体 API 为记住的设备提供了额外的操作。

1. ListDevices并AdminListDevices返回用户的设备密钥及其元数据的列表。
2. GetDevice并AdminGetDevice返回单个设备的设备密钥和元数据。
3. UpdateDeviceStatus并将用户的设备AdminUpdateDeviceStatus设置为已记住或未记住。
4. ForgetDevice并从用户的个人资料中AdminForgetDevice移除已确认的设备。

名称以 `Admin` 开头的 API 操作用于服务器端应用程序，必须使用 IAM 凭证进行授权。有关更多信息，请参阅 了解 API、OIDC 和托管登录页面身份验证。

获取设备密钥

每当用户使用用户群体 API 登录并且身份验证参数中未包含设备密钥作为 `DEVICE_KEY` 时，Amazon Cognito 都会在响应中返回新的设备密钥。在公共客户端应用程序中，将设备密钥放在应用程序存储中，以便您可以将其包含在将来的请求中。在机密服务器端应用程序中，使用用户的设备密钥设置浏览器 Cookie 或其它客户端令牌。

应用程序必须确认设备密钥并提供其它信息，然后用户才能使用其可信设备登录。向 Amazon Cognito 生成ConfirmDevice请求，使用设备密钥、友好名称、密码验证器和盐来确认用户的设备。如果您将用户群体配置为选择加入设备身份验证，Amazon Cognito 会在响应您的 `ConfirmDevice` 请求时，提示用户必须选择是否记住当前设备。在UpdateDeviceStatus请求中使用用户选择的内容进行回应。

当您确认用户的设备但未将其设置为记住的设备时，Amazon Cognito 会存储关联，但在您提供设备密钥时继续进行非设备登录。设备可以生成对用户安全和故障排除非常有用的日志。已确认但未记住的设

备不会利用登录特征，但会利用安全监控日志特征。当您为应用程序客户端激活高级安全特征并将设备占用空间编码到请求中时，Amazon Cognito 会将用户事件与已确认的设备关联起来。

获取新的设备密钥

1. 使用 InitiateAuthAPI 请求开始用户的登录会话。
2. 在收到标记用户登录会话完成的 JSON Web 令牌 (JWTs) RespondToAuthChallenge之前，使用回复所有身份验证挑战。
3. 在应用程序中，记录 Amazon Cognito 在其 `RespondToAuthChallenge` 或 `InitiateAuth` 响应的 `NewDeviceMetadata` 中返回的值：`DeviceGroupKey` 和 `DeviceKey`。
4. 为用户生成新的 SRP 密钥：盐和密码验证程序。此功能在提供 SRP 库 SDKs 的中可用。
5. 提示用户输入设备名称，或根据用户的设备特征生成一个名称。
6. 在 ConfirmDeviceAPI 请求中提供用户的访问令牌、设备密钥、设备名称和 SRP 密钥。如果用户群体设置为始终记住设备，则用户的注册已完成。
7. 如果 Amazon Cognito 对于 `ConfirmDevice` 响应了 `"UserConfirmationNecessary": true`，请提示您的用户选择是否要记住该设备。如果他们确认要记住设备，请使用用户的访问令牌、设备密钥和生成 UpdateDeviceStatusAPI 请求`"DeviceRememberedStatus": "remembered"`。
8. 如果您已指示 Amazon Cognito 记住该设备，那么当用户下次登录时，看到的不是 MFA 质询，而是 `DEVICE_SRP_AUTH` 质询。

使用设备登录

将用户的设备配置为记住后，当用户使用相同的设备密钥登录时，Amazon Cognito 不再要求用户提交 MFA 代码。设备身份验证仅用设备身份验证质询取代 MFA 身份验证质询。您不能仅使用设备身份验证登录用户。用户必须首先使用其密码或自定义质询完成身份验证。以下是在记住的设备上对用户进行身份验证的过程。

要在使用自定义身份验证质询 Lambda 触发器的流程中执行设备身份验证，请在您的 InitiateAuthAPI `DEVICE_KEY` 请求中传递参数。在用户成功完成所有质询并且 `CUSTOM_CHALLENGE` 质询返回的 `issueTokens` 值为 `true` 之后，Amazon Cognito 将返回一个最终 `DEVICE_SRP_AUTH` 质询。

使用设备登录

1. 从客户端存储中检索用户的设备密钥。
2. 使用 InitiateAuthAPI 请求开始用户的登录会话。选择一个 `AuthFlow`：`USER_SRP_AUTH`、`REFRESH_TOKEN_AUTH`、`USER_PASSWORD_AUTH` 或

CUSTOM_AUTH。在 AuthParameters 中，将用户的设备密钥添加到 DEVICE_KEY 参数中，并包括所选登录流程所需的其它参数。

a. 您还可以在对身份验证质询的 PASSWORD_VERIFIER 响应的参数中传递 DEVICE_KEY。

3. 完成质询响应，直到您在响应中收到 DEVICE_SRP_AUTH 质询。
4. 在 RespondToAuthChallengeAPI 请求中，发送USERNAMEDEVICE_KEY、DEVICE_SRP_AUTH和ChallengeName的和参数SRP_A。
5. Amazon Cognito 以 DEVICE_PASSWORD_VERIFIER 质询进行响应。此质询响应包括 SECRET_BLOCK 和 SRP_B 的值。
6. 使用您的 SRP 库，生成并提交 PASSWORD_CLAIM_SIGNATURE、PASSWORD_CLAIM_SECRET_BLOCK、TIMESTAMP、USERNAME 和 DEVICE_KEY 参数。在其它 RespondToAuthChallenge 请求中提交这些内容。
7. 完成其他挑战，直到收到用户的挑战 JWTs。

以下伪代码演示如何计算 DEVICE_PASSWORD_VERIFIER 质询响应的值。要使用设备进行 SRP 身份验证，请为您的用户生成一个新的 SRP 密钥：一个新的高熵密码DeviceSecret、一个盐和相关的密码验证器。这些值不同于用于用户的 SRP 身份验证的密码、salt 和验证器。它们仅用于设备身份验证，并且仅存储在设备上。用于为用户设备生成 SRP 密钥的功能可在不同版本的 SRP 库中找到。SDKs

```
PASSWORD_CLAIM_SECRET_BLOCK = SECRET_BLOCK
TIMESTAMP = "Tue May 27 00:09:40 UTC 2025"
k = SHA256(N || g) as a non-negative integer in big-endian
u = SHA256(SRP_A || SRP_B) as a non-negative integer in big-endian
x = SHA256(salt || SHA256(DeviceGroupKey || DeviceKey || ":" || DeviceSecret)) as a
 non-negative integer in big-endian
S_USER = (SRP_B - k * g^x)^(a + u * x) % N
K_USER = HKDF_HMAC_SHA256(salt=u, ikm=S_USER, info="Caldera Derived Key", length=16
 bytes)
PASSWORD_CLAIM_SIGNATURE = Base64(HMAC_SHA256(key=K_USER, message=(DeviceGroupKey ||
 DeviceKey || PASSWORD_CLAIM_SECRET_BLOCK || TIMESTAMP)))
```

查看、更新和忘记设备

您可以使用 Amazon Cognito API 在应用程序中实现以下特征。

1. 显示有关用户的当前设备的信息。
2. 显示用户的所有设备的列表。

3. 忘记设备。
4. 更新设备记住的状态。

授权以下描述中的 API 请求的访问令牌必须包含 `aws.cognito.signin.user.admin` 范围。Amazon Cognito 会将此范围的声明添加到您使用 Amazon Cognito 用户群体 API 生成的所有访问令牌中。第三方 IdPs 必须为其向 Amazon Cognito 进行身份验证的用户单独管理设备和 MFA。在托管登录中，您可以请求`aws.cognito.signin.user.admin`范围，但是托管登录会自动将设备信息添加到高级安全用户日志中，并且不提供记住设备的功能。

显示有关设备的信息

您可以查询有关用户设备的信息，以确定设备当前是否仍在使用中。例如，您可能希望在记住的设备已有 90 天未登录后将其停用。

- 要在公共客户端应用程序中显示用户的设备信息，请在 GetDeviceAPI 请求中提交用户的访问密钥和设备密钥。
- 要在机密客户端应用程序中显示用户的设备信息，请使用 Amazon 凭据签署 AdminGetDeviceAPI 请求并提交用户的用户名、设备密钥和用户池。

显示用户的所有设备的列表

您可以显示用户的所有设备及其属性的列表。例如，您可能要验证当前设备是否与记住的设备相匹配。

- 在公共客户端应用程序中，在 ListDevicesAPI 请求中提交用户的访问令牌。
- 在机密客户端应用程序中，使用 Amazon 凭据签署 AdminListDevicesAPI 请求并提交用户的用户名和用户池。

忘记设备

您可以删除用户的设备密钥。当您确定您的用户不再使用设备时，或者当您检测到异常活动并希望提示用户再次完成 MFA 时，您可能需要这样做。要稍后再次注册设备，必须生成并存储新的设备密钥。

- 在公共客户端应用程序中，在 ForgetDeviceAPI 请求中提交用户的设备密钥和访问令牌。
- 在机密客户端应用程序中，在 AdminForgetDeviceAPI 请求中提交用户的设备密钥和访问令牌。

作用域、M2M 以及 APIs 带资源服务器的场景

为用户池配置域后，Amazon Cognito 会自动配置 OAuth 2.0 授权服务器和托管 Web UI，其中包含您的应用程序可以向用户展示的注册和登录页面。有关更多信息，请参阅 用户池托管登录。您可以选择希望授权服务器添加到访问令牌中的范围。范围授权访问资源服务器和用户数据。

资源服务器是一个 OAuth 2.0 的 API 服务器。为了保护受访问权限保护的资源，它会验证用户群体中的访问令牌所包含的范围是否授权所请求的方法和它所保护的 API 中的路径。它根据令牌签名验证发放者，根据令牌到期时间验证有效性，并根据令牌声明中的范围验证访问级别。用户池范围位于访问令牌 `scope` 声明中。有关 Amazon Cognito 访问令牌中的声明的更多信息，请参阅了解访问令牌。

借助 Amazon Cognito，访问令牌中的范围可以授权访问外部属性 APIs 或用户属性。您可以向本地用户、联合用户或计算机身份发放访问令牌。

主题

- API 授权
- Machine-to-machine (M2M) 授权
- 关于范围
- 关于资源服务器

API 授权

以下是您可以使用亚马逊 Cognito 令牌 APIs 授权请求的一些方法：

访问令牌

将 Amazon Cognito 授权方添加到 REST API 方法请求配置时，将授权范围添加到授权方配置中。使用此配置，您的 API 会接受 `Authorization` 标头中的访问令牌，并检查访问令牌中的可接受范围。

ID 令牌

当您在 REST API 中将有效的 ID 令牌传递给 Amazon Cognito 授权方时，API Gateway 会接受请求并将 ID 令牌内容传递给 API 后端。

Amazon Verified Permissions

在 Verified Permissions 中，您可以选择创建 API 关联策略存储。Verified Permissions 创建并分配一个 Lambda 授权方，该授权方处理请求 `Authorization` 标头中的 ID 令牌或访问令牌。此

Lambda 授权方将您的令牌传递到策略存储，然后 Verified Permissions 将其与策略进行比较，并将向授权方返回允许或拒绝决定。

更多资源

- 在 API Gateway 中控制和管理对 REST API 的访问
- 使用 Amazon Verified Permissions 进行授权

Machine-to-machine (M2M) 授权

Amazon Cognito 支持使用计算机身份访问 API 数据的应用程序。用户池中的计算机身份是在应用程序服务器上运行并连接到远程的机密客户端 APIs。其操作无需用户交互：计划任务、数据流或资产更新。当这些客户端使用访问令牌授权其请求时，它们会执行机器对机器（M2M）授权。在 M2M 授权中，共享密钥取代访问控制中的用户凭证。

通过 M2M 授权访问 API 的应用程序必须具有客户端 ID 和客户端密钥。在您的用户池中，必须构建支持客户端凭证授予的应用程序客户端。要支持客户端凭证，您的应用程序客户端必须具有客户端密钥，且您必须有用户池域。在此流程中，您的计算机身份直接从 令牌端点 请求访问令牌。对于客户端凭证授予，您只能在访问令牌中授权来自资源服务器的自定义范围。有关设置应用程序客户端的更多信息，请参阅特定于应用程序的应用程序客户端设置。

来自客户端凭证授予的访问令牌实际上是一个可验证的声明，表明您希望计算机身份向 API 请求哪些操作。要详细了解访问令牌如何授权 API 请求，请继续阅读。有关示例应用程序，请参阅使用 Amazon CDK 进行基于 Amazon Cognito 和 API Gateway 的机器到机器授权。

M2M 授权的计费模式不同于每月活跃用户 (MAUs) 的计费方式。用户身份验证会根据每个活跃用户的数量收取费用，而 M2M 计费则反映了活跃的客户端凭证应用程序客户端和令牌请求总量。有关更多信息，请参阅 Amazon Cognito 定价。要控制 M2M 授权的成本，请优化访问令牌的持续时间和应用程序发出的令牌请求数量。有关使用 API Gateway 缓存来减少 M2M 授权中的新令牌请求的方法，请参阅管理用户池令牌到期和缓存。

有关优化会增加 Amazon 账单成本的 Amazon Cognito 操作的信息，请参阅。管理 成本

machine-to-machine(M2M) 客户端凭证的客户端元数据

您可以在 M2M 请求中传递客户端元数据。客户机元数据是来自用户或应用程序环境的其他信息，这些信息可能有助于实现的结果令牌生成前 Lambda 触发器。在使用用户委托人的身份验证操作中，您可以将客户端元数据传递给 RespondToAuthChallengeAPI 请求正文中的令牌生成前触发器。AdminRespondToAuthChallenge由于应用程序通过直接向 M2M 请求来执行生成访问令牌的流

程令牌端点，因此它们的模型不同。在客户端凭证令牌请求的 POST 正文中，将带有客户端元数据对象 URL-encoded (`x-www-form-urlencoded`) 的`aws_client_metadata`参数传递给字符串。有关示例请求，请参阅具有基本授权的客户凭证。以下是传递键值对的示例参数。`{"environment": "dev", "language": "en-US"}`

```
aws_client_metadata=%7B%22environment%22%3A%20%22dev%22,%20%22language%22%3A%20%22en-US
%22%7D
```

关于范围

范围是应用程序可请求的对资源的访问权限的级别。在 Amazon Cognito 访问令牌中，范围由您与用户群体建立的信任提供支持：一个具有已知数字签名的可信访问令牌发放者。用户群体可以生成访问令牌，其范围可以证明您的客户可以管理自己的部分或全部用户个人资料，或者可以从后端 API 检索数据。Amazon Cognito 用户池使用用户池预留 API 范围、自定义范围和 OpenID Connect（OIDC）范围来发放访问令牌。

用户群体预留 API 范围

Amazon Cognito 用户池 API 中的 `aws.cognito.signin.user.admin` 范围授权当前用户执行自助操作。它授权访问令牌的持有者通过和 API 操作查询和更新有关该持有者的所有信息。GetUserUpdateUserAttributes当您使用 Amazon Cognito 用户群体 API 对用户进行身份验证时，这是您在访问令牌中收到的唯一范围。这也是您读写已授权应用程序客户端读写的用户属性所需的唯一范围。您也可以在发往 对端点授权 的请求中请求此范围。仅此范围不足以向 userInfo 端点 请求用户属性。对于同时授权用户群体 API 和 用户 `userInfo` 请求的访问令牌，您必须在一个 `/oauth2/authorize` 请求中同时请求 `openid` 和 `aws.cognito.signin.user.admin` 这两个范围。

自定义范围

自定义作用域授权向资源服务器保护的外部服务器发 APIs 出的请求。您可以使用其他类型的范围请求自定义范围。您可以在此页面中找到有关自定义范围的更多信息。

OpenID Connect（OIDC）范围

当您使用用户池授权服务器（包括托管登录）对用户进行身份验证时，必须请求范围。您可以在 Amazon Cognito 授权服务器中对用户群体本地用户和第三方联合用户进行身份验证。OIDC 范围授权您的应用从您的用户池的 userInfo 端点 中读取用户信息。通过该 OAuth 模型，您可以从`userInfo`端点查询用户属性，它可以针对大量用户属性请求优化您的应用程序。`userInfo` 端点返回权限级别的属性，该级别由访问令牌中的范围决定。您可以授权您的应用程序客户端颁发具有以下 OIDC 范围的访问令牌。

openid

OpenID Connect (OIDC) 查询的最小范围。授权 ID 令牌、唯一标识符声明 sub 以及请求其他范围的能力。

> **Note**
>
> 当您请求 openid 范围而不请求其他范围时，您的用户群体 ID 令牌和 userInfo 响应将包括您的应用程序客户端可以读取的所有用户属性的声明。当您同时请求 openid 和其他 OIDC 范围（例如 profile、email 和 phone）时，ID 令牌和 userInfo 响应的内容将受到其他范围的限制。
> 例如，如果发送到 对端点授权 的请求带有参数 scope=openid+email，则将返回带有 sub、email 和 email_verified 的 ID 令牌。来自此请求的访问令牌也将从 userInfo 端点 返回这些属性。带有参数 scope=openid 的请求将在 ID 令牌中返回所有客户端可以读取的属性，userInfo 响应也是如此。

配置文件

授权应用程序客户端可以读取的所有用户属性。

电子邮件

授权用户属性 email 和 email_verified。如果有已明确设置的值，Amazon Cognito 将返回 email_verified。

phone

授权用户属性 phone_number 和 phone_number_verified。

关于资源服务器

资源服务器 API 可能会授予对数据库中信息的访问权限，或者控制您的 IT 资源。亚马逊 Cognito 访问令牌可以授权访问 APIs 该支持 OAuth 2.0。亚马逊 API Gateway REST 内置 APIs 了对使用亚马逊 Cognito 访问令牌进行授权的支持。应用程序会将 API 调用中的访问令牌传递到资源服务器。资源服务器将检查访问令牌以确定是否应授予访问权限。

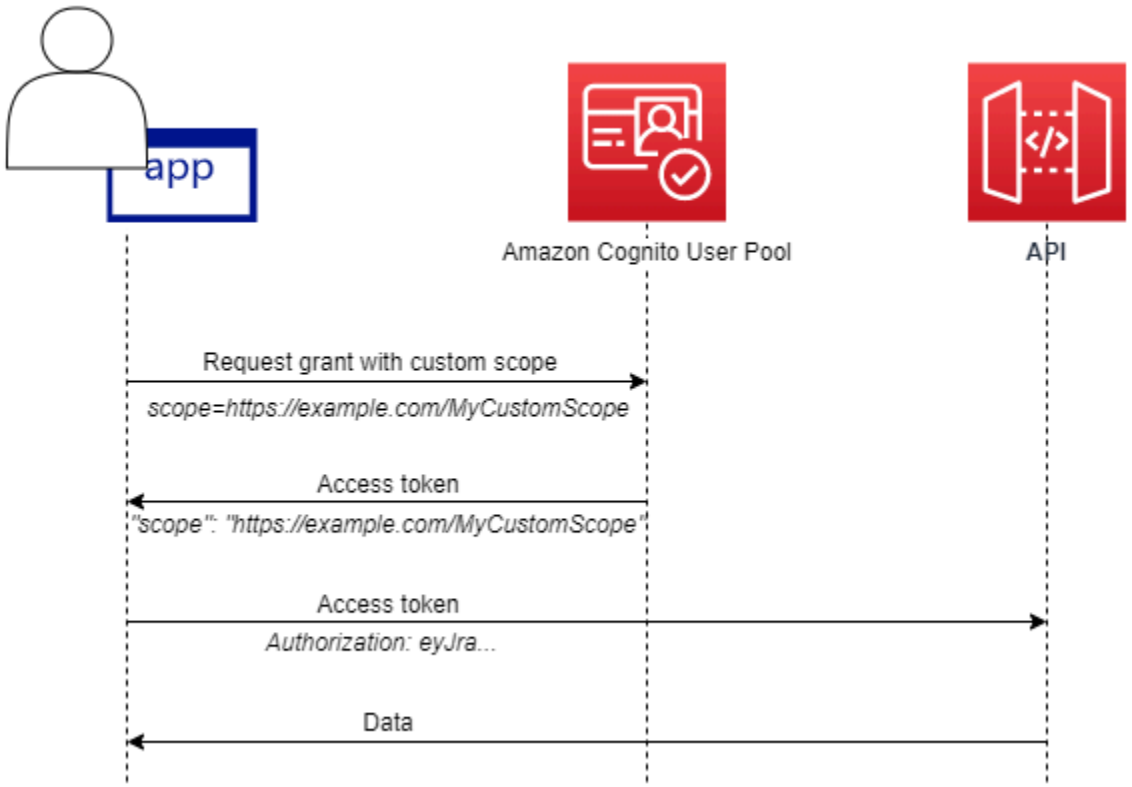
Amazon Cognito 将来可能会更新用户群体访问令牌的架构。如果您的应用程序在将访问令牌传递给 API 之前分析其内容，则您必须对代码进行设计以接受架构的更新。

自定义范围由您定义，它会扩展用户群体的授权功能，以包括与查询和修改用户及其属性无关的目的。例如，如果您有一个照片资源服务器，它可能会定义两个范围：`photos.read` 用于对照片的读取访问，`photos.write` 用于写入/删除访问。您可以配置 API 以接受用于授权的访问令牌，并授予 `HTTP GET` 请求使用 `scope` 声明中的 `photos.read` 访问令牌，以及授予 `HTTP POST` 请求使用 `photos.write` 访问令牌。这些是自定义范围。

> **Note**
>
> 您的资源服务器在处理访问令牌内的任何声明之前必须验证访问令牌的签名和到期日期。有关验证令牌的更多信息，请参阅验证 JSON 网络令牌。有关在 Amazon API Gateway 中验证和使用用户群体令牌的更多信息，请参阅博客将 Amazon Cognito 用户群体与 API Gateway 集成。API Gateway 是用于检查访问令牌和保护您的资源的一个很好的选择。有关 API Gateway Lambda 授权方的更多信息，请参阅使用 API Gateway Lambda 授权方。

概览

使用 Amazon Cognito，您可以创建 OAuth 2.0 资源服务器并将自定义范围与它们关联起来。访问令牌中的自定义范围可向 API 中的特定操作授权。您可以授权用户群体中的任何应用程序客户端从您的任何资源服务器发布自定义范围。将您的自定义范围与应用程序客户端关联，并从 OAuth 2.0 授权码授权、隐式授权和客户端凭证授权中请求这些范围令牌端点。Amazon Cognito 在访问令牌中将自定义范围添加到 `scope` 声明中。客户端可对其资源服务器使用访问令牌，然后服务器基于令牌中给出的范围做出授权决定。有关访问令牌范围的更多信息，请参阅将令牌与用户池结合使用。



要获得具有自定义范围的访问令牌，您的应用程序必须向 令牌端点 发出请求以兑换授权代码或请求客户端凭证授予。在托管登录中，您还可以从隐式授权中请求访问令牌中的自定义范围。

> **Note**
>
> 因为它们是为以用户池作为 IdP 的人机交互身份验证而设计的 InitiateAuth，AdminInitiateAuth并且请求仅在访问令牌中生成具有单`scope`一值的声明。`aws.cognito.signin.user.admin`

管理资源服务器和自定义范围

在创建资源服务器时，您必须提供资源服务器名称和资源服务器标识符。对于您在资源服务器中创建的每个范围，您都必须提供范围名称和描述。

- 资源服务器名称：资源服务器的易记名称，如 `Solar system object tracker` 或 `Photo API`。
- 资源服务器标识符：资源服务器的唯一标识符。标识符是您希望与 API 关联的任何名称，例如 `solar-system-data`。您可以配置更长的标识符，例如 `https://solar-system-data-`

api.example.com，作为对 API URI 路径的更直接引用，但较长的字符串会增加访问令牌的大小。

- 范围名称：scope 声明中需要的值。例如 sunproximity.read。
- 描述：范围的友好描述。例如 Check current proximity to sun。

Amazon Cognito 可以在任何用户的访问令牌中包含自定义范围，无论这些用户是用户群体的本地用户还是与第三方身份提供者的联合身份验证用户。在身份验证流程中，您可以使用包含托管登录的 OAuth 2.0 授权服务器为用户的访问令牌选择范围。您的用户的身份验证必须从以 scope 作为请求参数之一的对端点授权开始。以下是推荐的资源服务器格式。对于标识符，请使用 API 友好名称。对于自定义范围，请使用它们授权的操作。

```
resourceServerIdentifier/scopeName
```

例如，您在柯伊伯带发现了一颗新的小行星，您想通过 solar-system-data API 对其进行注册。授权对小行星数据库进行写操作的范围是 asteroids.add。当您请求授权您注册发现的小行星的访问令牌时，请将 scope HTTPS 请求参数格式设置为 scope=solar-system-data/asteroids.add。

从资源服务器中删除一个范围不会删除其与所有客户端的关联。而是范围标记为非活动。Amazon Cognito 不会为访问令牌添加非活动的范围，但如果您的应用程序请求访问令牌，则会正常进行。如果您稍后再次将范围添加到资源服务器，则 Amazon Cognito 会再次将其写入访问令牌。如果您请求的范围尚未与应用程序客户端关联，则无论您是否将其从用户群体资源服务器中删除，身份验证都会失败。

您可以使用 Amazon Web Services Management Console、API 或 CLI 为用户池定义资源服务器和范围。

为您的用户池定义资源服务器（Amazon Web Services Management Console）

您可以使用为您的用户池定义资源服务器。 Amazon Web Services Management Console

定义资源服务器

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 User Pools（用户池），然后选择要编辑的用户池。
3. 选择 “品牌” 下的 “域” 菜单，然后找到资源服务器。
4. 选择 Create a resource server（创建资源服务器）。
5. 输入 Resource server name（资源服务器名称）。例如 Photo Server。

6. 输入 Resource server identifier（资源服务器识符）。例如 `com.example.photos`。
7. 输入您的资源的 Custom scopes（自定义范围），例如 `read` 和 `write`。
8. 对于每个 Scope name（范围名称），输入一个 Description（描述），如 `view your photos` 和 `update your photos`。
9. 选择创建。

可以在资源服务器下的 “域” 菜单的 “自定义范围” 列中查看您的自定义范围。可以从 “应用程序” 下的 “应用程序客户端” 菜单中为应用程序客户端启用自定义范围。选择应用程序客户端，找到 “登录” 页面，然后选择 “编辑”。添加 Custom scopes（自定义范围），然后选择 Save changes（保存更改）。

为您的用户池（Amazon CLI 和 Amazon API）定义资源服务器

使用以下命令可为您的用户池指定资源服务器设置。

创建资源服务器

- Amazon CLI: `aws cognito-idp create-resource-server`
- Amazon API: CreateResourceServer

获取有关您的资源服务器设置的信息

- Amazon CLI: `aws cognito-idp describe-resource-server`
- Amazon API: DescribeResourceServer

列出用户池的所有资源服务器的相关信息

- Amazon CLI: `aws cognito-idp list-resource-servers`
- Amazon API: ListResourceServers

删除资源服务器

- Amazon CLI: `aws cognito-idp delete-resource-server`
- Amazon API: DeleteResourceServer

更新资源服务器的设置

- Amazon CLI: `aws cognito-idp update-resource-server`
- Amazon API: UpdateResourceServer

使用 Amazon Pinpoint 进行用户池分析

> **Note**
>
> 终止支持通知： Amazon 将于 2026 年 10 月 30 日终止对亚马逊 Pinpoint 的支持。2026 年 10 月 30 日之后，您将无法再访问亚马逊 Pinpoint 控制台或亚马逊 Pinpoint 资源（终端节点、区段、活动、旅程和分析）。有关更多信息，请参阅 Amazon Pinpoint 终止支持。注意： APIs 与短信相关、语音、移动推送、OTP 和电话号码验证不受此更改的影响，并受 Amazon 最终用户消息的支持。

Amazon Pinpoint 用户群体与 Amazon Pinpoint 集成，为 Amazon Cognito 用户群体提供分析并丰富 Amazon Pinpoint 活动的用户数据。Amazon Pinpoint 提供分析和有针对性的市场活动，以使用推送通知推动用户与移动应用程序的交互。借助 Amazon Cognito 用户池中的 Amazon Pinpoint 分析支持，您可以在 Amazon Pinpoint 控制台中跟踪用户池注册、登录、身份验证失败、每日活跃用户 DAUs () 和每月活跃用户 () MAUs。您可以深入查看不同日期范围或属性的数据，例如设备平台、设备区域设置和应用程序版本。

您还可以为应用程序设置自定义属性。然后可以使用这些属性在 Amazon Pinpoint 上对用户进行细分，并向他们发送有针对性的推送通知。如果您在 Amazon Cognito 控制台的应用程序客户端菜单中的应用程序客户端分析配置中选择与 Amazon Pinpoint 共享用户属性数据，Amazon Pin p oint 会为用户电子邮件地址和电话号码创建额外的终端节点。

当您使用 Amazon Cognito 控制台在用户群体中激活 Amazon Pinpoint 分析时，您还会创建一个服务相关角色，Amazon Cognito 在向 Amazon Pinpoint 发出针对用户群体的 API 请求时代入该角色。添加您的分析配置的 IAM 委托人必须拥有CreateServiceLinkedRole权限。服务相关角色是。AWSServiceRoleForAmazonCognitoIdp有关更多信息，请参阅 对 Amazon Cognito 使用服务相关角色。

当您在 Amazon Cognito API 中向应用程序客户端应用 `AnalyticsConfiguration` 时，您可以为 Amazon Pinpoint 分配自定义 IAM 角色和外部 ID 来代入该角色。此角色必须信任 `cognito-idp` 服务主体，如果角色信任策略需要外部 ID，则它必须与您的 `AnalyticsConfiguration` 相匹配。您必须针对 Amazon Pinpoint 项目向该角色授予 `cognito-idp:Describe*` 权限和以下权限。

- `mobiletargeting:UpdateEndpoint`
- `mobiletargeting:PutEvents`

Amazon Cognito 和 Amazon Pinpoint 区域可用性

下表显示了满足以下条件之一的 Amazon Cognito 和 Amazon Pinpoint 之间的 Amazon Web Services 区域 映射。

- 您只能在美国东部（弗吉尼亚州北部）（us-east-1）区域使用 Amazon Pinpoint 项目。
- 您可以在相同的区域或者在美国东部（弗吉尼亚州北部）（us-east-1）区域使用 Amazon Pinpoint 项目

默认情况下，Amazon Cognito 只能向位于相同 Amazon Web Services 区域中的 Amazon Pinpoint 项目发送分析。此规则的例外情况是下表中的区域，以及 Amazon Pinpoint 不可用的区域。

Amazon Pinpoint 未在以下区域推出。这些地区的 Amazon Cognito 用户群体不支持分析。

- 欧洲地区（米兰）
- 中东（巴林）
- 亚太地区（大阪）
- 以色列（特拉维夫）
- 非洲（开普敦）
- 亚太地区（雅加达）
- 亚太地区（马来西亚）

表中显示了您构建 Amazon Cognito 用户群体的区域与 Amazon Pinpoint 中对应区域的关系。您必须在可用区域中配置 Amazon Pinpoint 项目，才能将其与 Amazon Cognito 集成。

Amazon Cognito 用户群体区域	Amazon Pinpoint 项目所在区域
ap-northeast-1	us-east-1
ap-northeast-2	us-east-1
ap-south-1	us-east-1、ap-sou1

Amazon Cognito 用户群体区域	Amazon Pinpoint 项目所在区域
ap-southeast-1	us-east-1
ap-southeast-2	us-east-1、ap-southeast-2
ca-central-1	us-east-1
eu-central-1	us-east-1、eu-central-1
eu-west-1	us-east-1、eu-west-1
eu-west-2	us-east-1
us-east-1	us-east-1
us-east-2	us-east-1
us-west-2	us-east-1、us-west-2

区域映射示例

- 如果您在 ap-northest-1 中创建用户群体，则可以在 us-east-1 中创建您的 Amazon Pinpoint 项目。
- 如果您在 ap-south-1 中创建用户群体，则可以在 us-east-1 或 ap-south-1 中创建您的 Amazon Pinpoint 项目。

> **Note**
>
> 除了上表中的项目 Amazon Web Services 区域 外，Amazon Cognito 只能使用与您的用户池位于同一区域的 Amazon Pinpoint 项目。如果 Amazon Pinpoint 在您构建用户群体的区域中不可用且未在表中列出，那么 Amazon Cognito 在该区域不支持 Amazon Pinpoint 分析。有关详细的 Amazon Web Services 区域 信息，请参阅 Amazon Pinpoint 端点和配额。

指定 Amazon Pinpoint 分析设置（Amazon Web Services Management Console）

您可以配置 Amazon Cognito 用户群体以向 Amazon Pinpoint 发送分析数据。Amazon Cognito 仅为本地用户将分析数据发送到 Amazon Pinpoint。将您的用户群体配置为与 Amazon Pinpoint 项目关联后，

您必须在 API 请求中包含 `AnalyticsMetadata`。有关更多信息，请参阅 将您的应用程序与 Amazon Pinpoint 集成。

指定分析设置

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择 User Pools（用户群体）并从列表中选择一个现有的用户群体。
3. 选择应用程序客户端菜单，然后选择要更新的应用程序客户端。
4. 在 Pinp oint 分析下方的分析选项卡中，选择启用。
5. 选择 Pinpoint Region（Pinpoint 区域）。
6. 选择 Amazon Pinpoint project（Amazon Pinpoint 项目）或者选择 Create Amazon Pinpoint project（创建 Amazon Pinpoint 项目）。

   > **Note**
   >
   > Amazon Pinpoint 项目 ID 是 Amazon Pinpoint 项目特有的由 32 个字符组成的字符串。它在 Amazon Pinpoint 控制台中列出。
   > 您可以将多个 Amazon Cognito 应用程序映射到单个 Amazon Pinpoint 项目。但是，每个 Amazon Cognito 应用程序只能映射到一个 Amazon Pinpoint 项目。
   > 在 Amazon Pinpoint 中，每个项目都应该是单个应用程序。例如，如果游戏开发员有两款游戏，每款游戏应该是单独的 Amazon Pinpoint 项目，即使这两款游戏使用同一 Amazon Cognito 用户池。有关 Amazon Pinpoint 项目的更多信息，请参阅在 Amazon Pinpoint 中创建项目。

7. 在 User data sharing（用户数据共享）下，如果您希望 Amazon Cognito 发送电子邮件地址和电话号码到 Amazon Pinpoint 并为用户创建额外的端点，则选择 Share user data with Amazon Pinpoint（与 Amazon Pinpoint 共享用户数据）。用户验证其电子邮件地址和电话号码后，Amazon Cognito 只在它们可用于用户账户时才会与 Amazon Pinpoint 共享。

   > **Note**
   >
   > 端点唯一地标识可以使用 Amazon Pinpoint 向其发送推送通知的用户设备。有关端点的更多信息，请参阅《Amazon Pinpoint 开发人员指南》中的添加端点。

8. 选择保存更改。

指定亚马逊 Pinpoint 分析设置（Amazon CLI 和 Amazon API）

使用以下命令为您的用户池指定 Amazon Pinpoint 分析设置。

在创建应用程序时为用户池的现有客户端应用程序指定分析设置

- Amazon CLI: `aws cognito-idp create-user-pool-client`
- Amazon API: CreateUserPoolClient

为用户池的现有客户端应用程序更新分析设置

- Amazon CLI: `aws cognito-idp update-user-pool-client`
- Amazon API: UpdateUserPoolClient

> **Note**
>
> 在您使用`ApplicationArn`时，Amazon Cognito 支持区域内集成

将您的应用程序与 Amazon Pinpoint 集成

您可以在用户群体 API 中针对 Amazon Cognito 本地用户将分析元数据发布到 Amazon Pinpoint。

本地用户

注册了账户或在您的用户群体中创建的用户，而不是通过第三方身份提供者 (IdP) 登录的用户。

用户群体 API

您可以使用带有自定义用户界面 (UI) 的应用程序与 S Amazon DK 集成的操作。对于通过托管登录登录的联合用户或本地用户，您无法传递分析元数据。有关用户群体 API 操作的列表，请参阅 Amazon Cognito API 参考。

在您将用户群体配置为发布到活动后，Amazon Cognito 会将以下 API 操作的元数据传递给 Amazon Pinpoint。

- `AdminInitiateAuth`
- `AdminRespondToAuthChallenge`

- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `ForgotPassword`
- `InitiateAuth`
- `ResendConfirmationCode`
- `RespondToAuthChallenge`
- `SignUp`

要将有关用户会话的元数据传递到 Amazon Pinpoint 活动，请在 API 请求的 `AnalyticsMetadata` 参数中包含 `AnalyticsEndpointId` 值。 JavaScript 例如，请参阅为什么我的 Amazon Cognito 用户池分析没有显示在我的 Amazon Pinpoint 控制面板上？ 在Amazon 知识中心中。

Amazon Cognito 用户池的电子邮件设置

应用程序中的某些事件可能导致 Amazon Cognito 向用户发送电子邮件。例如，如果您将用户池配置为需要电子邮件验证，则当用户在应用程序中注册新账户或重置其密码时，Amazon Cognito 会发送电子邮件。根据发起电子邮件递送的操作，电子邮件将包含验证码或临时密码。

为处理电子邮件递送，您可以使用以下任一选项：

- 内置于 Amazon Cognito 服务中的默认电子邮件配置。
- 您的 Amazon Simple Email Service (Amazon SES) 配置。

创建用户群体后，您可以更改传递选项。

Amazon Cognito 会向您的用户发送电子邮件，其中包含用户可以输入的代码或用户可以选择的 URL 链接。下表显示了可以生成电子邮件的事件。

消息选项

活动	API 操作	传递选项	格式选项	可自定义	消息模板
忘记密码	ForgotPassword, AdminResetUserPassword	电子邮件、短信	code	否	不适用

活动	API 操作	传递选项	格式选项	可自定义	消息模板
邀请	AdminCreateUser	电子邮件、短信	code	是	邀请消息
自行注册	SignUp, ResendConfirmationCode	电子邮件、短信	代码，链接	是	验证消息
电子邮件地址或电话号码验证	UpdateUserAttributes, AdminUpdateUserAttributes, GetUserAttributeVerificationCode	电子邮件、短信	code	是	验证消息
多重身份验证 (MFA)	AdminInitiateAuth, InitiateAuth	电子邮件[1]、短信、身份验证器应用程序	code	是[2]	MFA 消息

[1] 需要高级安全特征和 Amazon SES 电子邮件配置。

[2] 用于短信和电子邮件消息。

Amazon SES 会对电子邮件收费。有关详情，请参阅 Amazon SES 定价。

要了解有关电子邮件 MFA 的更多信息，请参阅 短信和电子邮件消息 MFA。

Amazon Cognito 可能会在短时间内阻止向单个目的地发送额外的电子邮件或短信。如果您认为自己的用户群受到了影响，请配置并查看日志中是否存在消息传送错误，然后联系您的客户团队。

默认电子邮件配置

Amazon Cognito 可以使用其默认电子邮件配置为您处理电子邮件递送。当您使用默认选项时，Amazon Cognito 会限制您的用户池每天可以发送的电子邮件数量。有关服务限制的更多信息，请参阅 Amazon Cognito 中的限额。对于典型的生产环境，默认的电子邮件限制低于所需的递送量。要启用更高的送达量，您可使用您的 Amazon SES 电子邮件配置。

当您使用默认配置时，您将使用 Amazon 管理的 Amazon SES 资源来发送电子邮件消息。Amazon SES 将返回查无此人的邮件的电子邮件地址添加到账户级禁止列表或全局禁止列表。如果无法送达的电子邮件地址稍后变为可送达，则当您的用户池配置为使用默认配置时，您无法控制将其从禁止列表中删除的行为。电子邮件地址可以无限期地保留在 Amazon管理的禁止列表中。要管理无法送达的电子邮件地址，请将 Amazon SES 电子邮件配置与账户级别的禁止列表一起使用，如下一节中所述。

使用默认电子邮件配置时，您可以使用以下任一电子邮件地址作为 FROM 地址：

- 默认电子邮件地址 no-reply@verificationemail.com。
- 自定义电子邮件地址。在可以使用您自己的电子邮件地址之前，您必须向 Amazon SES 验证此地址，并且向 Amazon Cognito 授予使用此地址的权限。

Amazon SES 电子邮件配置

您的应用程序需要的递送量可能高于默认选项所提供的递送量。要增加可能的递送量，请将您的 Amazon SES 资源与用户池一起使用来向用户发送电子邮件。当您使用自己的 Amazon SES 配置发送电子邮件消息时，您还可以监控您的电子邮件发送活动。

在可以使用您的 Amazon SES 配置之前，您必须向 Amazon SES 验证一个或多个电子邮件地址或域。将经验证的电子邮件地址或已验证域的地址，用作分配给用户池的 FROM 电子邮件地址。当 Amazon Cognito 向您的用户发送电子邮件时，它会以您的名义调用 Amazon SES 并使用您的电子邮件地址。

当您使用 Amazon SES 配置时，以下条件适用：

- 您的用户池的电子邮件传送限制与适用于您的 Amazon Web Services 账户中的 Amazon SES 验证电子邮件地址的限制相同。
- 您可以使用 Amazon SES 中覆盖全局禁止列表的账户级禁止列表来管理发送到无法送达的电子邮件地址的邮件。当您使用账户级禁止列表时，退回的电子邮件消息会影响您的账户作为发件人的声誉。有关更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的使用 Amazon SES 账户级禁止列表。

Amazon SES 电子邮件配置区域

Amazon Web Services 区域 在 Amazon SES 中配置电子邮件时，创建用户池的位置有三个要求之一。您可以从与您的用户池相同的区域、多个区域（包括相同区域），或者一个或多个远程区域通过 Amazon SES 发送电子邮件消息。为了获得出色性能，在您可以选择的情况下，在与您的用户池相同的区域中，通过经 Amazon SES 验证的身份发送电子邮件。

经过 Amazon SES 验证的身份的区域要求类别

仅限区域内

您的用户池可以像用户池 Amazon Web Services 区域 一样发送身份经过验证的电子邮件。在没有自定义 FROM 电子邮件地址的默认电子邮件配置中，Amazon Cognito 使用同一区域中的 no-reply@verificationemail.com 已验证身份。

向后兼容

您的用户池可以在相同 Amazon Web Services 区域 或以下备用区域之一发送身份经过验证的电子邮件：

- 美国东部（弗吉尼亚州北部）
- 美国西部（俄勒冈州）
- 欧洲地区（爱尔兰）

此特征支持您可能在 Amazon Cognito 服务启动时为了满足当时的要求而创建的用户池资源的连续性。该时期的用户池只能在有限的数量内发送身份经过验证的电子邮件 Amazon Web Services 区域。在没有自定义 FROM 电子邮件地址的默认电子邮件配置中，Amazon Cognito 使用同一区域中的 no-reply@verificationemail.com 已验证身份。

备用区域

您的用户池可以在用户池区域之外的备用 Amazon Web Services 区域 地址发送身份经过验证的电子邮件。当 Amazon SES 在提供 Amazon Cognito 的区域不可用时，就会出现这种配置。

Amazon SES 在备用区域发送经过验证的身份的授权策略时，必须信任来源区域的 Amazon Cognito 服务主体。有关更多信息，请参阅 授予权限以使用默认的电子邮件配置。

在其中一些区域，对于默认的 COGNITO_DEFAULT 电子邮件配置，Amazon Cognito 在两个备用区域之间拆分电子邮件消息。在这些情况下，要使用自定义 FROM 电子邮件地址，Amazon SES 在每个备用区域发送经过验证的身份的授权策略时，必须信任来源区域的 Amazon Cognito 服务主体。有关更多信息，请参阅 授予权限以使用默认的电子邮件配置。借助这些区域中的 DEVELOPER Amazon SES 电子邮件配置，您必须在第一个列出的区域中使用经过验证的身份，并将其配置为信任用户池区域中的 Amazon Cognito 服务主体。例如，在中东（阿联酋）的用户池中，将欧洲地区（法兰克福）经过验证的身份配置为信任 cognito-idp.me-central-1.amazonaws.com。在没有自定义 FROM 电子邮件地址的默认电子邮件配置中，Amazon Cognito 使用每个区域中的 no-reply@verificationemail.com 已验证身份。

> **Note**
>
> 在以下条件组合下，必须在 Region 元素中EmailConfiguration使用通配符指定SourceArn参数，格式为arn:*${Partition}*:ses:*:*${Account}*:identity/*${IdentityName}*。这允许您的用户池发送两者中都具有相同经过验证的身份 Amazon Web Services 账户 的电子邮件 Amazon Web Services 区域。
>
> - 你的 EmailSendingAccount 是COGNITO_DEFAULT。
> - 您想使用自定义 FROM 地址。
> - 您的用户池在备用区域发送电子邮件。
> - 您的用户池在后面的 Amazon SES 支持区域表中指定了第二个 [1] 备用区域。

如果您使用软件开发工具包、Amazon Cognito API 或 CLI、或以编程方式创建用户池，则您的用户池将 Amazon CDK使用参数 Amazon CloudFormation为您的用户池指定的 Amazon SES 身份发送电子邮件。 Amazon SourceArn EmailConfigurationAmazon SES 身份必须占据支持 Amazon Web Services 区域的。如果您的 EmailSendingAccount 是 COGNITO_DEFAULT 而且您没有指定 SourceArn 参数，则 Amazon Cognito 使用您创建用户池所在区域中的资源，从 no-reply@verificationemail.com 发送电子邮件。

下表显示了您可以在 Amazon Cognito 上使用 Amazon SES 身份 Amazon Web Services 区域 的地方。

用户池区域	区域选项	Amazon SES 支持的区域
美国东部（弗吉尼亚州北部）	向后兼容	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
美国东部（俄亥俄州）	向后兼容	美国东部（俄亥俄）、美国东部（弗吉尼亚北部）、欧洲（爱尔兰）
美国西部（加利福尼亚北部）	仅限区域内	美国西部（加利福尼亚北部）

用户池区域	区域选项	Amazon SES 支持的区域
美国西部（俄勒冈州）	向后兼容	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
加拿大（中部）	向后兼容	加拿大（中部）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
加拿大西部（卡尔加里）	备用区域	加拿大（中部）、美国西部（北加利福尼亚）[1]
亚太地区（东京）	向后兼容	亚太地区（东京）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（香港）	备用区域	亚太地区（新加坡）、亚太地区（东京）[1]
亚太地区（首尔）	向后兼容	亚太地区（首尔）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（马来西亚）	备用区域	亚太地区（悉尼）、亚太地区（新加坡）[1]
亚太地区（孟买）	向后兼容	亚太地区（孟买）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（海得拉巴）	备用区域	亚太地区（孟买）、亚太地区（新加坡）[1]
亚太地区（新加坡）	向后兼容	亚太地区（新加坡）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）

用户池区域	区域选项	Amazon SES 支持的区域
亚太地区（悉尼）	向后兼容	亚太地区（悉尼）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
亚太地区（大阪）	仅限区域内	亚太地区（大阪）
亚太地区（雅加达）	仅限区域内	亚太地区（雅加达）
亚太地区（墨尔本）	备用区域	亚太地区（悉尼）、亚太地区（新加坡）[1]
欧洲地区（爱尔兰）	向后兼容	美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲地区（伦敦）	向后兼容	欧洲（伦敦）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲地区（巴黎）	仅限区域内	欧洲地区（巴黎）
欧洲地区（法兰克福）	向后兼容	欧洲（法兰克福）、美国东部（弗吉尼亚北部）、美国西部（俄勒冈）、欧洲（爱尔兰）
欧洲（苏黎世）	备用区域	欧洲地区（法兰克福）、欧洲地区（伦敦）[1]
欧洲地区（斯德哥尔摩）	仅限区域内	欧洲地区（斯德哥尔摩）
欧洲地区（米兰）	仅限区域内	欧洲地区（米兰）
欧洲（西班牙）	备用区域	欧洲地区（巴黎）、欧洲地区（斯德哥尔摩）[1]
中东（巴林）	仅限区域内	中东（巴林）

用户池区域	区域选项	Amazon SES 支持的区域
中东（阿联酋）	备用区域	欧洲地区（法兰克福）、欧洲地区（伦敦）[1]
南美洲（圣保罗）	仅限区域内	南美洲（圣保罗）
以色列（特拉维夫）	仅限区域内	以色列（特拉维夫）
非洲（开普敦）	仅限区域内	非洲（开普敦）

[1] 在具有默认电子邮件配置的用户池中使用。Amazon Cognito 在每个区域中通过具有相同电子邮件地址的经验证身份来分发电子邮件。要使用自定义 FROM 地址，请使用格式为 `arn:${Partition}:ses:*:${Account}:identity/${IdentityName}` 的 `SourceArn` 参数来配置 `EmailConfiguration`。

为您的用户池配置电子邮件

完成以下步骤为用户池配置电子邮件设置。根据您要使用的设置，您可能需要 Amazon SES、Amazon Identity and Access Management （IAM）和 Amazon Cognito 中的 IAM 权限。

> **Note**
>
> 您在这些步骤中创建的资源无法跨 Amazon Web Services 账户进行共享。例如，您不能为一个账户中的用户池配置用户池，然后将其用于另一个账户中的 Amazon SES 电子邮件地址。如果您在多个账户中使用 Amazon Cognito，请为每个账户中重复这些步骤。

步骤 1：使用 Amazon SES 验证电子邮件地址或域

在配置您的用户池之前，如果您要执行以下任一操作，则必须使用 Amazon SES 验证一个或多个电子邮件地址或域：

- 使用您自己的电子邮件地址作为 FROM 地址
- 使用您的 Amazon SES 配置处理电子邮件送达

通过验证您的电子邮件地址或域，您确认您拥有该电子邮件地址，这有助于防止未经授权的使用。

有关使用 Amazon SES 验证电子邮件地址的更多信息，请参阅 Amazon Simple Email Service 开发人员指南中的验证电子邮件地址。有关使用 Amazon SES 验证域的信息，请参阅验证域。

步骤 2：将您的账户移出 Amazon SES 沙盒

如果您使用的是默认 Amazon Cognito 电子邮件配置，则跳过此步骤。

当你第一次在任何地方使用 Amazon SES 时 Amazon Web Services 区域，它会将你置 Amazon Web Services 账户 于该地区的 Amazon SES 沙箱中。Amazon SES 使用沙盒防止欺诈和滥用。如果您使用您的 Amazon SES 配置来处理电子邮件送达，则必须将您的 Amazon Web Services 账户 移出沙盒，然后 Amazon Cognito 才能向用户发送电子邮件。

在沙盒中，Amazon SES 会对您可以发送的电子邮件数量和可以发送电子邮件的位置施加限制。您可以仅向已通过 Amazon SES 验证的地址和域发送电子邮件，也可以将其发送到 Amazon SES 邮箱模拟器地址。在您 Amazon Web Services 账户 仍处于沙箱状态时，请不要将您的 Amazon SES 配置用于生产中的应用程序。在这种情况下，Amazon Cognito 无法将邮件发送到您用户的电子邮件地址。

要将您 Amazon Web Services 账户 从沙箱中移除，请参阅《亚马逊简单电子邮件服务开发者指南》中的移出 Amazon SES 沙箱。

步骤 3：授予 Amazon Cognito 电子邮件权限

您可能需要向 Amazon Cognito 授予特定权限，然后它才能向您的用户发送电子邮件。您授予的权限以及用于授予权限的过程取决于您使用的是默认电子邮件配置还是您的 Amazon SES 配置。

授予权限以使用默认的电子邮件配置

仅当您将用户池配置为使用 Cognito 发送电子邮件或将 `EmailSendingAccount` 设置为 `COGNITO_DEFAULT` 时，才能完成此步骤。

使用默认的电子邮件配置，您的用户池可以使用以下任一地址发送电子邮件。

- 默认地址 `no-reply@verificationemail.com`。
- 来自您在 Amazon SES 中的已验证电子邮件地址或域的自定义 FROM 地址。

如果您使用自定义地址，Amazon Cognito 需要额外权限，以便使用此地址向您的用户发送电子邮件。通过 Amazon SES 中地址或域的发送授权策略来授予这些权限。如果您使用 Amazon Cognito 控制台向您的用户池添加自定义地址，此策略会自动附加到 Amazon SES 验证的电子邮件地址。但是，如果您在控制台之外配置用户池，例如使用 Amazon CLI 或 Amazon Cognito API，则必须使用 A mazon SES 控制台或 API 附加策略。PutIdentityPolicy

> **Note**
>
> 您只能使用 Amazon CLI 或 Amazon Cognito API 在已验证的域中配置 FORM 地址。

发送授权策略根据使用 Amazon Cognito 调用 Amazon SES 的账户资源，来允许或拒绝访问。有关基于资源的策略的更多信息，请参阅 IAM 用户指南。在 Amazon SES 开发人员指南中可以找到基于资源的策略示例。

Example 发送授权策略

以下示例发送授权策略授予 Amazon Cognito 使用经 Amazon SES 验证的身份的有限能力。Amazon Cognito 在代表 `aws:SourceArn` 中的用户池和 `aws:SourceAccount` 条件中的账户时才能发送电子邮件。

Regions with Amazon SES

用户池区域或备用区域中的发送授权策略必须允许 Amazon Cognito 服务主体发送电子邮件消息。有关更多信息，请参阅区域表。如果您的用户池区域与 Amazon SES 区域中的至少一个值匹配，请在以下示例中使用全局服务主体配置您的发送授权策略。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "stmnt1234567891234",
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "email.cognito-idp.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
```

```
                    "aws:SourceArn": "<your user pool ARN>"
                }
            }
        }
    ]
}
```

Opt-in Regions without Amazon SES

在可用 Amazon Cognito Amazon Web Services 区域 的地方，Amazon SES 并非在所有可选模式中都可用。中东（阿联酋）是一个例子，只能在欧洲地区（法兰克福）（eu-central-1）使用经过验证的身份发送电子邮件。在使用默认电子邮件配置的用户池中，Amazon Cognito 也会在这两个区域中的每一个使用经过验证的身份发送电子邮件消息。就中东（阿联酋）而言，新增区域是欧洲地区（伦敦）。您必须更新两个区域中的发送授权策略。

每个备用区域中的发送授权策略必须允许用户池选择加入区域中的 Amazon Cognito 服务主体发送电子邮件消息。有关更多信息，请参阅区域表。如果您的区域标记为备用区域，请使用区域服务主体配置您的发送授权策略，如下例所示。根据需要将示例区域标识*me-central-1*符替换为所需的区域 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": [
                    "cognito-idp.me-central-1.amazonaws.com"
                ]
            },
            "Action": [
                "SES:SendEmail",
                "SES:SendRawEmail"
            ],
            "Resource": "<your SES identity ARN>",
            "Condition": {
                "StringEquals": {
                    "aws:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "aws:SourceArn": "<your user pool ARN>"
                }
```

```
            }
        }
    ]
}
```

有关策略语法的更多信息，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略。

有关更多示例，请参阅《Amazon Simple Email Service 开发人员指南》中的 Amazon SES 发送授权策略示例。

授予权限以使用您的 Amazon SES 配置

如果您将用户池配置为使用您的 Amazon SES 配置，Amazon Cognito 在向用户发送电子邮件时，需要额外的权限代表您调用 Amazon SES。此授权将通过 IAM 服务授予。

当您使用此选项配置用户池时，Amazon Cognito 将创建一个服务相关角色，这是您 Amazon Web Services 账户中的一种 IAM 角色类型。此角色包含允许 Amazon Cognito 访问 Amazon SES 并使用您的地址发送电子邮件的权限。

Amazon Cognito 使用设置配置的用户会话的 Amazon 凭证创建您的服务相关角色。此会话的 IAM 权限必须包括 `iam:CreateServiceLinkedRole` 操作。有关 IAM 中权限的更多信息，请参阅 I AM 用户指南中的Amazon 资源访问管理。

有关 Amazon Cognito 创建的服务相关角色的更多信息，请参阅对 Amazon Cognito 使用服务相关角色。

步骤 4：配置用户池

如果您要将您的用户池配置为使用以下内容，请完成以下步骤：

- 自定义 FROM 地址（显示为电子邮件发件人）。
- 自定义 REPLY-TO 地址，用于接收您的用户发送到您的 FROM 地址的邮件
- Amazon SES 配置

> **Note**
>
> 如果您的已验证身份是一个电子邮件地址，Amazon Cognito 会默认将该电子邮件地址设置为 FROM 和 REPLY-TO 电子邮件地址。但如果您的已验证身份是一个域，则必须为 FROM 电子邮件地址提供一个值。

如果您要使用默认 Amazon Cognito 电子邮件配置和地址，则跳过此过程。

配置用户池以使用自定义电子邮件地址

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “身份验证方法” 菜单，找到 “电子邮件配置”，选择 “编辑”。
5. 在 Edit email configuration（编辑电子邮件配置）页面中，选择 Send email from Amazon SES（使用 Amazon SES 发送电子邮件）或 Send email with Amazon Cognito（使用 Amazon Cognito 发送电子邮件）。仅当您选择 Send email from Amazon SES（使用 Amazon SES 发送电子邮件）时，您才可以自定义 SES Region（SES 区域）、Configuration Set（配置集） 和 FROM sender name（FROM 发件人姓名）。
6. 要使用自定义 FROM 地址，请完成以下步骤：
   a. 在 SES region（SES 区域）下，选择包含验证的电子邮件地址的区域。
   b. 在 FROM email address（FROM 电子邮件地址）下，选择您的电子邮件地址。使用您已通过 Amazon SES 验证的电子邮件地址。
   c. （可选）在 Configuration set（配置集）下，选择 Amazon SES 使用的配置集。进行此更改并保存可创建服务相关角色。
   d. （可选）在 FROM sender address（FROM 发件人地址）下，输入电子邮件地址。您可以仅提供电子邮件地址，也可以同时提供电子邮件地址和格式为 `Jane Doe <janedoe@example.com>` 的易记名称。
   e. （可选）在 REPLY-TO email address（REPLY-TO 电子邮箱地址）下，输入要用来接收用户发送到您的 FROM 地址的邮件的电子邮件地址。
7. 选择 Save changes（保存更改）。

相关主题

- 自定义电子邮件验证消息
- 自定义用户邀请消息

Amazon Cognito 用户池的短信设置

您的用户池的某些 Amazon Cognito 事件可能会导致 Amazon Cognito 向您的用户发送短信。例如，如果您将用户池配置为需要电话验证，则当用户在应用程序中注册新账户或重置其密码时，Amazon Cognito 会发送短信。根据发起短信的操作，短信中将包含验证码、临时密码或欢迎消息。

Amazon Cognito 使用 Amazon Simple Notification Service (Amazon SNS) 传送短信。如果这是您首次通过 Amazon Cognito 或 Amazon SNS 发送短信，Amazon SNS 会将您放在沙盒环境。在沙盒环境中，您可以对应用程序的 SMS 文本消息进行测试。在沙盒中，只能将消息发送给经过验证的电话号码。

Amazon SNS 对短信收取费用。有关更多信息，请参阅 Amazon SNS 定价。

Amazon Cognito 会向您的用户发送带有他们可以输入的验证码的短信。下表显示了可以生成短信的事件。

消息选项

活动	API 操作	传递选项	格式选项	可自定义	消息模板
忘记密码	ForgotPassword, AdminResetUserPassword	电子邮件、短信	code	否	不适用
邀请	AdminCreateUser	电子邮件、短信	code	是	邀请消息
自行注册	SignUp, ResendConfirmationCode	电子邮件、短信	代码，链接	是	验证消息
电子邮件地址或电话号码验证	UpdateUserAttributes, AdminUpdateUserAttributes, GetUserAt	电子邮件、短信	code	是	验证消息

活动	API 操作	传递选项	格式选项	可自定义	消息模板
	tributeVerificationCode				
多重身份验证 (MFA)	AdminInitiateAuth, InitiateAuth	短信、身份验证器应用程序	code	是[1]	MFA 消息

[1] 用于短信。

Amazon SNS 会对短信收费。有关更多信息，请参阅 Amazon SNS 定价。

要了解有关 MFA 的更多信息，请参阅 短信和电子邮件消息 MFA。

Amazon Cognito 可能会在短时间内阻止向单个目的地发送额外的电子邮件或短信。如果您认为自己的用户群受到了影响，请配置并查看日志中是否存在消息传送错误，然后联系您的客户团队。

最佳实践

由于全球范围内未经请求的短信流量巨大，一些政府在短信发送者和接收者之间设置了障碍。当您使用短信进行 MFA 和用户更新时，必须采取额外的步骤来确保您的短信已送达。您还必须监控用户可能居住的国家/地区的 SMS-message-related法规，并保持您的 SMS 消息配置处于最新状态。有关更多信息，请参阅《Amazon Simple Notification Service 开发人员指南》中的移动文本消息（SMS）。

使用短信对用户进行身份验证和验证不是安全最佳做法。电话号码可能会变更所有者，而可能无法可靠地代表您拥有的 用户 MFA 要素。而是在应用程序中或使用第三方 IdP 实现 TOTP MFA。您还可以使用自定义身份验证质询 Lambda 触发器创建其他自定义身份验证要素。

查看以下链接，了解有关保护 SMS 消息传送架构的信息。

- 使用 Amazon Cognito 用户池降低用户注册欺诈和短信激增的风险
- 防范短信抽水：新 Amazon 功能可帮助抵御人为膨胀的流量

首次在 Amazon Cognito 用户池中设置 SMS 消息

Amazon Cognito 使用 Amazon SNS 向您的用户池发送短信。您还可以使用自定义 SMS 发件人 Lambda 触发器，通过自己的资源发送 SMS 消息。首次将 Amazon SNS 设置为在特定 Amazon Web Services 区域地区发送短信时，Amazon SNS 会将 Amazon Web Services 账户 您置于该地区的短信

沙箱中。Amazon SNS 使用沙箱来防止欺诈和滥用行为并满足合规要求。当你进入沙箱 Amazon Web Services 账户 时，Amazon SNS 会施加一些限制。例如，您最多可以向 10 个已通过 Amazon SNS 验证的电话号码发送短信。在您 Amazon Web Services 账户 仍处于沙箱状态时，请勿将您的 Amazon SNS 配置用于生产中的应用程序。当您位于沙盒中时，Amazon Cognito 无法向用户的电话号码发送消息。

向用户池用户发送 SMS 文本消息

1. 准备一个 IAM 角色，Amazon Cognito 可以使用该角色通过 Amazon SNS 发送 SMS 消息
2. 选择 Amazon SNS 短信 Amazon Web Services 区域
3. 获取源身份以将 SMS 消息发送到美国电话号码
4. 确认您位于 SMS 沙盒中
5. 将您的账户移出 Amazon SNS 沙盒
6. 在 Amazon SNS 中验证 Amazon Cognito 的电话号码
7. 在 Amazon Cognito 中完成用户群体设置

准备一个 IAM 角色，Amazon Cognito 可以使用该角色通过 Amazon SNS 发送 SMS 消息

当您从用户群体发送 SMS 消息时，Amazon Cognito 将代入您的账户中的 IAM 角色。Amazon Cognito 使用分配给该角色的 `sns:Publish` 权限向您的用户发送 SMS 消息。在 Amazon Cognito 控制台中，您可以从用户池的 “身份验证方法” 菜单的 “短信” 下设置 IAM 角色选择，也可以在用户池创建向导期间进行此选择。

以下示例 IAM 角色信任策略授予 Amazon Cognito 用户群体有限代入角色的能力。只有在满足以下条件时 Amazon Cognito 才可以代入该角色：

- 代入角色操作是代表 `aws:SourceArn` 条件下的用户池来执行。
- 代入角色操作是代表通过 `aws:SourceAccount` 条件设置的 Amazon Web Services 账户 中的用户池来执行。
- 代入角色操作在 `sts:externalId` 条件中包括外部 ID。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "",
            "Effect": "Allow",
```

```
            "Principal": {
                "Service": "cognito-idp.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "sts:ExternalId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
                    "aws:SourceAccount": "111122223333"
                },
                "ArnLike": {
                    "aws:SourceArn": "arn:aws:cognito-idp:us-
west-2:111122223333:userpool/us-west-2_EXAMPLE"
                }
            }
        }
    ]
}
```

您可以在 `aws:SourceArn` 条件的值中指定准确的用户群体 ARN 或通配符 ARN。使用 DescribeUserPoolAPI 请求在 ARNs Amazon Web Services Management Console 或中查找您的用户池。

要发送用于多重身份验证的短信消息，您的 IAM 角色信任策略必须有一个 `sts:ExternalId` 条件。此条件的值必须与您的用户池SmsConfiguration的`ExternalId`属性相匹配。当您在 Amazon Cognito 控制台中创建用户池的过程中创建 IAM 角色时，Amazon Cognito 会在角色和用户池设置中为您配置外部 ID。使用现有的 IAM 角色时，情况并非如此。

您必须更新 UpdateUserPoolAPI 请求中的用户池`ExternalId`参数，并使用相同值的`sts:externalId`条件更新 IAM 角色信任策略。要了解如何使用 API 来更新用户池，同时保持原有配置不变，请参阅 更新用户池和应用程序客户端配置。

有关 IAM 角色和信任策略的更多信息，请参阅《Amazon Identity and Access Management 用户指南》中的角色术语和概念。

选择 Amazon SNS 短信 Amazon Web Services 区域

在某些情况下 Amazon Web Services 区域，您可以选择包含要用于发送 Amazon Cognito 短信的 Amazon SNS 资源的区域。在任何提供 Amazon Cognito Amazon Web Services 区域 的地方，除了亚太地区（首尔）之外，您都可以在创建用户池的地方使用 Amazon SNS 资源。 Amazon Web Services 区域 在有多个区域可供选择时，为了使您的 SMS 消息收发更快且更可靠，请使用与您的用户池位于相同区域中的 Amazon SNS 资源。

> **Note**
>
> 在中 Amazon Web Services Management Console，只有切换到新的 Amazon Cognito 控制台体验后，您才能更改短信资源的区域。

在新建用户池向导的 Configure message delivery（配置消息传输）步骤中，为 SMS 资源选择区域。您也可以在现有用户池的 “身份验证方法” 菜单中选择 “短信” 下的 “编辑”。

对于某些人来说，在发布时 Amazon Web Services 区域，Amazon Cognito使用其他地区的 Amazon SNS资源发送了短信。要设置您的首选区域，请使用用户池SmsConfigurationType对象的`SnsRegion`参数。当您通过下表以编程方式在 Amazon Cognito 区域中创建 Amazon Cognito 用户池资源并且您不提供 `SnsRegion` 参数时，您的用户池可以使用旧 Amazon SNS 区域中的 Amazon SNS 资源发送 SMS 消息。

亚太地区（首尔）的 Amazon Cognito 用户池 Amazon Web Services 区域 必须使用您在亚太地区（东京）地区的 Amazon SNS 配置。

Amazon SNS 将所有新账户的支出配额设定为每月 1.00 美元 (USD)。在使用 Amazon Cognito 时 Amazon Web Services 区域 ，你可能已经提高了支出限额。在更改 Amazon SNS 短信的配额之前，请在 Amazon 支持中心提出增加配额的案例，以提高您在新区域的限额。 Amazon Web Services 区域有关更多信息，请参阅 Amazon Simple Notification Service 开发人员指南中的请求对 Amazon SNS 提升您的每月 SMS 支出配额。

您可以使用相应 Amazon SNS 区域中的 Amazon SNS 资源为下表中的任何 Amazon Cognito 区域发送 SMS 消息。

Amazon Cognito 区域	Amazon SNS 区域
美国东部（俄亥俄州）	美国东部（俄亥俄）、美国东部（弗吉尼亚北部）
美国东部（弗吉尼亚州北部）	美国东部（弗吉尼亚北部）
美国西部（北加利福尼亚）	美国西部（加利福尼亚北部）
美国西部（俄勒冈州）	美国西部（俄勒冈州）
加拿大（中部）	加拿大（中部）、美国东部（弗吉尼亚北部）

Amazon Cognito 区域	Amazon SNS 区域
加拿大西部（卡尔加里）	加拿大西部（卡尔加里）
欧洲地区（法兰克福）	欧洲（法兰克福）、欧洲（爱尔兰）
欧洲地区（伦敦）	欧洲（伦敦）、欧洲（爱尔兰）
欧洲地区（爱尔兰）	欧洲地区（爱尔兰）
欧洲地区（巴黎）	欧洲地区（巴黎）
欧洲地区（斯德哥尔摩）	欧洲地区（斯德哥尔摩）
欧洲地区（米兰）	欧洲地区（米兰）
欧洲（西班牙）	欧洲（西班牙）
欧洲（苏黎世）	欧洲（苏黎世）
亚太地区（马来西亚）	亚太地区（新加坡）
亚太地区（孟买）	亚太地区（孟买）、亚太地区（新加坡）
亚太地区（海得拉巴）	亚太地区（海得拉巴）
亚太地区（香港）	亚太地区（新加坡）
亚太地区（首尔）	亚太地区（东京）
亚太地区（新加坡）	亚太地区（新加坡）
亚太地区（悉尼）	亚太地区（悉尼）
亚太地区（东京）	亚太地区（东京）
亚太地区（雅加达）	亚太地区（雅加达）
亚太地区（大阪）	亚太地区（大阪）
亚太地区（墨尔本）	亚太地区（墨尔本）

Amazon Cognito 区域	Amazon SNS 区域
中东（巴林）	中东（巴林）
中东（阿联酋）	中东（阿联酋）
南美洲（圣保罗）	南美洲（圣保罗）
以色列（特拉维夫）	以色列（特拉维夫）
非洲（开普敦）	非洲（开普敦）

获取源身份以将 SMS 消息发送到美国电话号码

无论您是构建 SMS 沙盒测试环境还是生产环境，如果您计划向美国电话号码发送短信，则必须获取源身份。

自 2021 年 6 月 1 日起，美国运营商要求提供源身份才能向美国电话号码发送短信。如果您没有源身份，则必须获取一个。请参阅《Amazon Pinpoint 用户指南》中的申请号码了解如何获取源身份。

如果您从事以下操作 Amazon Web Services 区域，则必须开 Amazon Web Services 支持 票才能获得发件人身份。有关说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的针对 SMS 消息收发请求支持。

- 美国东部（俄亥俄州）
- 欧洲地区（斯德哥尔摩）
- 欧洲地区（巴黎）
- 欧洲地区（米兰）
- Middle East (Bahrain)
- 南美洲（圣保罗）
- 美国西部（加利福尼亚北部）

当您的同一个来源身份有多个时 Amazon Web Services 区域，Amazon SNS 会按以下优先顺序选择来源身份类型：短码、10DLC、免费电话号码。无法更改此优先级。有关更多信息，请参阅 Amazon SNS FAQs。

确认您位于 SMS 沙盒中

按照以下过程确认您是否在 SMS 沙盒中。对每个有正式版 Amazon Cognito 用户池 Amazon Web Services 区域 的地方重复此操作。

在 Amazon Cognito 控制台中查看 SMS 沙盒状态

确认您位于 SMS 沙盒中

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入 Amazon 凭证。
2. 选择 User Pools（用户池）。
3. 从列表中选择现有用户池。
4. 选择 “身份验证方法” 菜单。
5. 在 SMS configuration（SMS 配置）部分，展开 Move to Amazon SNS production environment（迁移到 Amazon SNS 生产环境）。如果您的账户位于 SMS 沙盒中，您将看到以下消息：

   ```
   You are currently in the SMS Sandbox and cannot send SMS messages to
   unverified numbers.
   ```

   如果您没有看到此消息，则表明有人已经在您的账户中设置了 SMS 消息。跳至在 Amazon Cognito 中完成用户群体设置。
6. 选择消息中的 Amazon SNS 链接。这将在新选项卡中打开 Amazon SNS 控制台。
7. 验证您是否位于沙盒环境中。控制台消息会显示您的沙箱状态 Amazon Web Services 区域，如下所示：

   ```
   This account is in the SMS sandbox in US East (N. Virginia).
   ```

将您的账户移出 Amazon SNS 沙盒

如果正在测试应用程序，并且只需向管理员可以验证的电话号码发送 SMS 消息，请跳过此步骤。

要在生产环境中使用您的应用程序，请将账户移出 SMS 沙盒并放入生产环境。在中配置了包含您希望 Amazon Cognito 使用的 Amazon SNS 资源的原始身份后，您可以在留在 SMS 沙箱中时 Amazon Web Services 账户 验证美国电话号码。 Amazon Web Services 区域 当您的 Amazon SNS 环境投入生产时，您无需在 Amazon SNS 中验证用户电话号码即可向用户发送 SMS 消息。

有关详细说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的移出 SMS 沙盒。

在 Amazon SNS 中验证 Amazon Cognito 的电话号码

如果您已将账户移出 SMS 沙盒，则跳过此步骤。

当您位于 SMS 沙盒中时，您可以使用 Amazon SNS 向任何已验证的号码发送消息。

要验证电话号码，请执行以下操作：

1. 在 Amazon SNS 控制台的 Text messaging (SMS)（文本消息 (SMS)）部分，添加 Sandbox destination phone number（沙盒目标电话号码）。
2. 在您提供的电话号码上接收带有密码的 SMS 消息。
3. 在 Amazon SNS 控制台上，输入 SMS 消息中的 Verification code（验证代码）。

有关详细说明，请参阅《Amazon Simple Notification Service 开发人员指南》中的在 SMS 沙盒中添加并验证电话号码。

> **Note**
>
> 当您在 SMS 沙盒中时，Amazon SNS 限制可以验证的目标电话号码的数量。请参阅《Amazon Simple Notification Service 开发人员指南》中的 SMS 沙盒。

在 Amazon Cognito 中完成用户群体设置

返回您创建或编辑用户池的浏览器选项卡。完成过程。当您成功将 SMS 配置添加到用户群体后，Amazon Cognito 会向内部电话号码发送测试消息，以验证您的配置有效。Amazon SNS 会对每条测试 SMS 消息收费。

使用 Amazon Cognito 用户池安全功能

您可能想要保护应用程序，防止受到网络入侵、密码猜测、用户模拟以及恶意注册和登录的侵害。Amazon Cognito 用户池安全特征的配置是安全架构中的一个关键组件。您的应用程序的安全性是客户责任 “云端安全”，如责任 Amazon 共担模型中所述。本章中的工具有助于使您的应用程序安全设计符合这些目标。

在配置用户池时，必须作出的一项重要决定是，是否允许公开注册和登录。某些用户池选项（例如机密客户端、以管理员身份创建用户和确认，以及没有域的用户池）受到互联网攻击影响的程度较小。而

一个常见的使用场景是公共客户端，它们接受互联网上任何人的注册并将所有操作直接发送到您的用户池。在任何配置中（尤其是在这些公共配置中），我们建议在规划和部署用户池时要考虑安全特征。当不受欢迎的来源创建新的活跃用户或试图利用现有用户时，安全性不足也会影响您的 Amazon 账单。

MFA 和威胁防护适用于本地用户。第三方 IdPs 应对联合用户的安全状况负责。

用户池安全特征

多重身份验证 (MFA)

请求用户池通过电子邮件（使用 Essentials 或 Plus 功能计划）、短信或身份验证器应用程序发送的验证码，以确认用户池登录。

威胁防护

监控登录以了解风险指标，并应用 MFA 或阻止登录。向访问令牌添加自定义声明和范围。通过电子邮件发送 MFA 代码。

Amazon WAF 网页 ACLs

检查用户池端点和身份验证 API 的传入流量，以了解网络层和应用程序层是否存在不想要的活动。

区分大小写

防止创建除了字符大小写有所不同之外，其电子邮件地址或首选用户名与另一个用户完全相同的用户。

删除保护

防止自动化系统意外删除您的用户池。需要在 Amazon Web Services Management Console中进一步确认删除用户池。

用户存在错误

防止泄露用户池中现有的用户名和别名。无论用户名是否有效，在身份验证失败时都返回一个通用错误。

主题

- 向用户池添加 MFA
- 具有威胁防护功能的高级安全性
- 将 Amazon WAF Web ACL 与用户池关联

- 用户池区分大小写
- 用户池删除保护
- 管理用户存在错误响应

向用户池添加 MFA

MFA 将您具有的某种 身份验证因素添加到您已知的 初始因素（通常是用户名和密码）中。您可以选择 SMS 短信、电子邮件或基于时间的一次性密码 (TOTP) 作为其他因素，以密码作为主要身份验证因素的用户登录。

多重身份验证（MFA）可提高应用程序中本地用户的安全性。对于联合用户，Amazon Cognito 会将所有身份验证过程委托给 IdP，并且不会为他们提供额外的身份验证因素。

> **Note**
>
> 即使您的用户池需要 MFA，新用户首次登录您的应用程序时，Amazon Cognito 也会发放 OAuth 2.0 令牌。您的用户首次登录时的第二个身份验证因素是他们对 Amazon Cognito 发送给他们的验证消息的确认。如果您的用户池要求使用 MFA，Amazon Cognito 会提示您的用户注册一个额外的登录因素，以便在第一次之后的每次登录尝试期间使用。

借助自适应身份验证，可以将用户池配置为响应增加的风险级别需要额外的身份验证因素。要向用户池添加自适应身份验证，请参阅 具有威胁防护功能的高级安全性。

将用户池的 MFA 设置为 `required` 时，所有用户都必须完成 MFA 才能登录。要登录，每个用户至少设置一个 MFA 安全因素。如果需要 MFA，则必须在用户入职中包含 MFA 设置，这样您的用户池才允许他们登录。

当您将 MFA 设置为必填项时，托管登录会提示用户设置 MFA。当您在用户池中将 MFA 设置为可选时，托管登录不会提示用户。要使用可选的 MFA，您必须在应用程序中构建一个界面，以提示用户选择他们需要设置 MFA，然后引导他们完成 API 输入以验证他们的额外登录因素。

主题

- 关于用户池 MFA 的注意事项
- 用户 MFA 首选项
- 用户运行时的 MFA 逻辑详情

- 为多因素身份验证配置用户池
- 短信和电子邮件消息 MFA
- TOTP 软件令牌 MFA

关于用户池 MFA 的注意事项

在设置 MFA 之前，请考虑以下情况：

- 用户可以拥有 MFA，也可以使用无密码因素登录。
  - 在支持一次性密码或密钥的用户池中，您不能将 MFA 设置为必填项。
  - 当您的用户池中需要 MFA `AllowedFirstAuthFactors` 时 `WEB_AUTHNEMAIL_OTP`，您无法添加、或`SMS_OTP`。在 Amazon Cognito 控制台中，您无法编辑基于选择的登录选项以包含无密码因素。
  - 当用户@@ 池中需要使用 MFA 时，基于选择的登录仅在所有应用程序客户端中提供优惠`PASSWORD`和`PASSWORD_SRP`考虑因素。有关用户名密码流程的更多信息，请参阅本指南身份验证一章使用永久密码和安全负载登录中的使用永久密码登录和。
  - 在 MFA 为可选用户池中，已配置 MFA 因子的用户只能在基于选择的登录中使用用户名密码身份验证流程登录。这些用户有资格使用所有基于客户的登录流程。
- 用户首选 MFA 方法会影响他们可用于恢复密码的方法。首选 MFA 方式为电子邮件的用户无法通过电子邮件接收密码重置代码。首选 MFA 方式为短信的用户无法通过短信接收密码重置代码。

  当用户不符合条件，无法使用首选密码重置方法时，您的密码恢复设置必须提供替代选项。例如，您的恢复机制可能将电子邮件列为第一优先选项，而电子邮件 MFA 可能是您的用户池中的一个选项。在这种情况下，添加短信消息账户恢复作为第二个选项，或者使用管理 API 操作为这些用户重置密码。

  的示例请求正文UpdateUserPool说明了当电子邮件密码重置不可用时，用户可以通过短信回退到恢复状态。`AccountRecoverySetting`
- 用户无法通过相同的电子邮件地址或电话号码接收 MFA 和密码重置码。如果他们使用电子邮件中的一次性密码 (OTPs) 进行 MFA，则必须使用 SMS 消息进行账户恢复。如果他们使用来 OTPs 自 SMS 消息的 MFA，则必须使用电子邮件进行账户恢复。在具有 MFA 的用户池中，如果用户有电子邮件地址的属性但没有电话号码，或者有电话号码但没有电子邮件地址，则他们可能无法完成自助密码恢复。

  要防止出现用户无法在使用此配置的用户池中重置密码的状态，请根据需要设置`email`和`phone_number`属性。或者，您可以设置在用户注册或管理员创建用户配置文件时始终

收集和设置这些属性的流程。当用户同时拥有这两个属性时，Amazon Cognito 会自动向目标发送密码重置代码，该代码不是用户的 MFA 因子。

- 当您在用户池中激活 MFA 并选择短信或电子邮件作为第二个因素时，您可以向尚未在 Amazon Cognito 中验证的电话号码或电子邮件属性发送消息。在您的用户完成 MFA 后，Amazon Cognito 会`phone_number_verified`将其或属性设置为。`email_verified` `true`
- 在五次尝试提交 MFA 代码均未成功后，Amazon Cognito 会开始如登录尝试失败时的锁定行为中所描述的指数超时锁定过程。
- 如果您的账户位于包含您的用户池的亚马逊简单通知服务 (Amazon SNS) Simple Notification Service 资源的短信沙箱中，则必须先验证亚马逊 SNS 中的电话号码，然后才能发送短信。Amazon Web Services 区域 有关更多信息，请参阅 Amazon Cognito 用户池的短信设置。
- 要使用威胁防护更改用户的 MFA 状态以响应检测到的事件，请在 Amazon Cognito 用户池控制台中激活 MFA 并将其设置为可选。有关更多信息，请参阅 具有威胁防护功能的高级安全性。
- 电子邮件和短信消息要求您的用户分别具有电子邮件地址和电话号码属性。您可以在用户池中将 `email` 或 `phone_number` 设置为必需的属性。在这种情况下，除非用户提供电话号码，否则他们无法完成注册。如果您未将这些属性设置为必需，但想要进行电子邮件或短信消息 MFA，则可以在用户注册时提示他们输入电子邮件地址或电话号码。妥善的做法是将用户池配置为自动向用户发送消息来验证这些属性。

  如果用户通过短信或电子邮件成功接收了临时验证码，并在 VerifyUserAttributeAPI 请求中返回了该验证码，则 Amazon Cognito 会将电话号码或电子邮件地址视为已验证。或者，您的团队可以设置电话号码，并使用执行 AdminUpdateUserAttributesAPI 请求的管理应用程序将其标记为已验证。
- 如果您已将 MFA 设置为必需且激活了多个身份验证因素，则 Amazon Cognito 会提示新用户选择他们想要使用的 MFA 因素。用户必须有电话号码才能设置短信消息 MFA，必须有电子邮件地址才能设置电子邮件消息 MFA。如果用户没有为任何基于消息的可用 MFA 定义属性，则 Amazon Cognito 会提示他们设置 TOTP MFA。选择 MFA 因子 (`SELECT_MFA_TYPE`) 和设置所选因子 (`MFA_SETUP`) 的提示是对InitiateAuth和 AdminInitiateAuthAPI 操作的质询响应。

用户 MFA 首选项

用户可以设置多个 MFA 因素。只能激活一个因素。您可以在用户池设置中或从用户提示为用户选择有效的 MFA 首选项。当用户池设置和他们自己的用户级设置满足以下条件时，用户池会提示用户输入 MFA 验证码：

1. 您可以在用户池中将 MFA 设置为可选或必需。
2. 用户具有有效的`email`或`phone_number`属性，或者已为 TOTP 设置了身份验证器应用程序。

3. 至少有一个 MFA 因素处于活动状态。
4. 一个 MFA 因素设置为首选。

用户池设置及其对 MFA 选项的影响

用户池的配置会影响用户可以选择的 MFA 方法。以下是一些影响用户设置 MFA 能力的用户池设置。

- 在 Amazon Cognito 控制台登录菜单的多重身份验证配置中，您可以将 MFA 设置为可选或必选，也可以将其关闭。与此设置的 API 等效项是`CreateUserPool``UpdateUserPool`、和的MfaConfiguration参数`SetUserPoolMfaConfig`。

  此外，在多重身份验证配置中，MFA 方法设置决定了用户可以设置的 MFA 因素。与该设置相当的 API 就是SetUserPoolMfaConfig操作。
- 在 “登录” 菜单的 “用户帐户恢复” 下，您可以配置用户池向忘记密码的用户发送消息的方式。用户的 MFA 方法不能与忘记密码代码的用户池传送方法相同。忘记密码传送方法的 API 参数是和的AccountRecoverySetting参数。`CreateUserPool` `UpdateUserPool`

  例如，当您的恢复选项为 “仅限电子邮件” 时，用户无法设置电子邮件 MFA。这是因为您无法在同一个用户池中启用电子邮件 MFA 并将恢复选项设置为 “仅限电子邮件”。当您将此选项设置为 “电子邮件”（如果可用），否则设置为 “短信” 时，电子邮件是优先恢复选项，但是当用户没有资格恢复电子邮件时，您的用户池可以回退到 SMS 消息。在这种情况下，用户可以将电子邮件 MFA 设置为首选，并且只有在尝试重置密码时才能收到 SMS 消息。
- 如果您只将一种 MFA 方法设置为可用，则无需管理用户 MFA 首选项。
- 激活的短信配置会自动使短信成为用户池中的可用 MFA 方法。

  用户池中包含您自己的 Amazon SES 资源的活跃电子邮件配置，以及 Essentials 或 Plus 功能计划，会自动使电子邮件成为用户池中可用的 MFA 方法。
- 在用户池中将 MFA 设置为必需时，用户无法启用或禁用任何 MFA 方法。您只能设置首选方法。
- 当您在用户池中将 MFA 设置为可选时，托管登录不会提示用户设置 MFA，但是当用户有首选 MFA 方法时，它会提示用户输入 MFA 代码。
- 在全功能模式下激活威胁防护并配置自适应身份验证响应时，MFA 在您的用户池中必须是可选的。自适应身份验证的其中一个响应选项是要求其登录尝试被评估为包含一定风险级别的用户进行 MFA。

  控制台 “注册” 菜单中的 “必填属性” 设置决定了用户是否必须提供电子邮件地址或电话号码才能注册您的应用程序。当用户具有相应属性时，电子邮件和短信将成为符合条件的 MFA 因素。`CreateUserPool` 的 Schema 参数根据需要设置属性。

- 当您在用户池中将 MFA 设置为必填且用户使用托管登录登录时，Amazon Cognito 会提示他们从用户池的可用方法中选择一种 MFA 方法。托管登录处理电子邮件地址或电话号码的收集以及 TOTP 的设置。下图演示了 Amazon Cognito 向用户提供的选项背后的逻辑。

为用户配置 MFA 首选项

您可以在具有访问令牌授权的自助服务模式中为用户配置 MFA 首选项，或者在具有管理 API 操作的管理员管理模式中为用户配置 MFA 首选项。这些操作启用或禁用 MFA 方法，并将多种方法之一设置为首选选项。在用户设置 MFA 首选项后，Amazon Cognito 会在登录时提示他们提供来自他们首选 MFA 方法的验证码。未设置首选项的用户会收到在 `SELECT_MFA_TYPE` 质询中选择首选方法的提示。

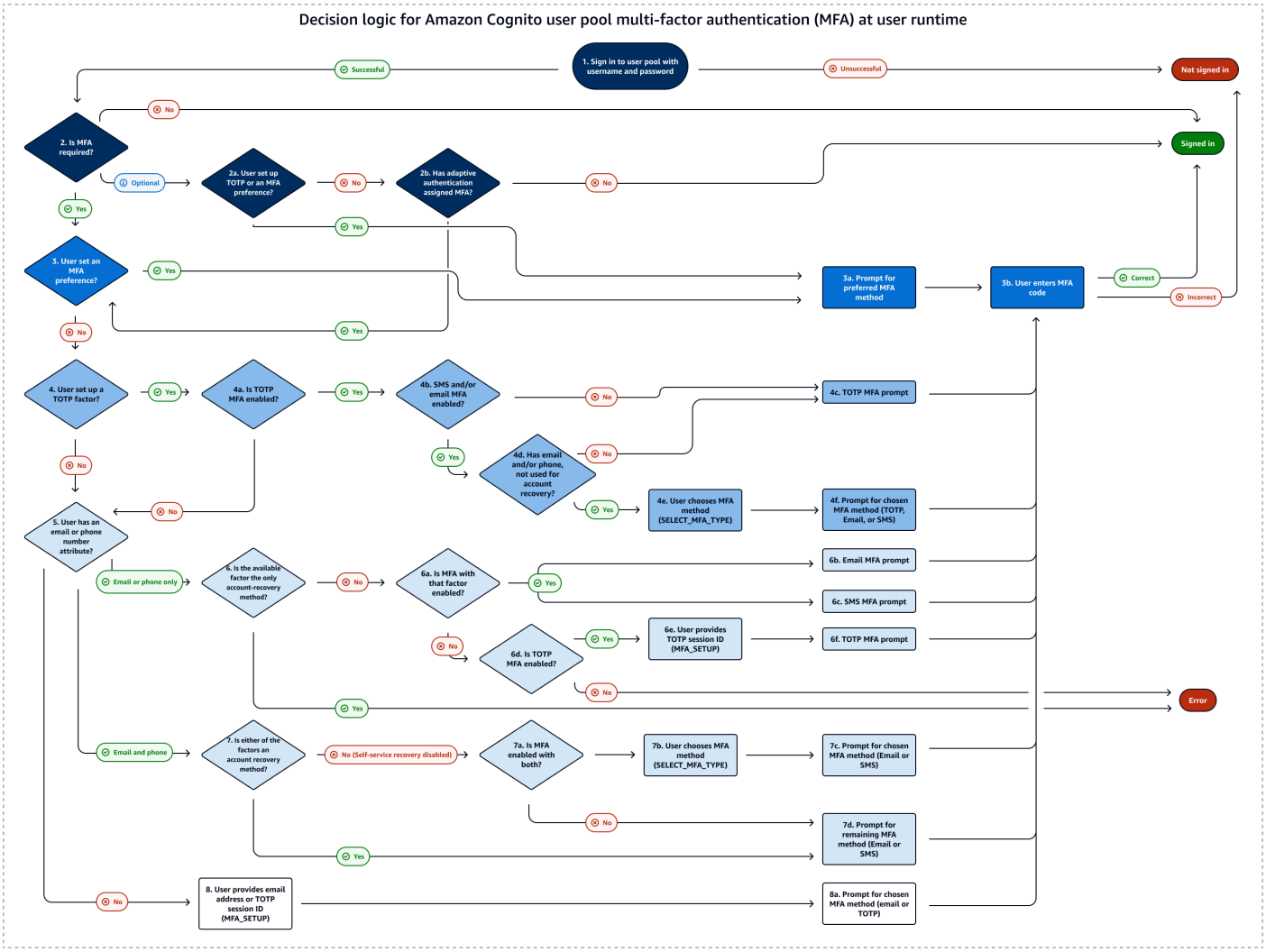
- 在用户自助服务模式或公共应用程序中 SetUserMfaPreference，使用登录用户的访问令牌进行授权，设置 MFA 配置。
- 在管理员管理的应用程序或机密应用程序中，使用管理 Amazon 凭证进行授权 AdminSetUserPreference，设置 MFA 配置。

您还可以从 Amazon Cognito 控制台的 “用户” 菜单中设置用户 MFA 首选项。有关 Amazon Cognito 用户池 API 中的公共和机密身份验证模式的更多信息，请参阅了解 API、OIDC 和托管登录页面身份验证。

用户运行时的 MFA 逻辑详情

为了确定用户登录时要采取的步骤，您的用户池会评估用户 MFA 首选项、用户属性、用户池 MFA 设置、威胁防护操作和自助账户恢复设置。然后，它会让用户登录，提示他们选择 MFA 方法，提示他们设置 MFA 方法，或者提示他们输入 MFA。要设置 MFA 方法，用户必须提供电子邮件地址或电话号码或注册 TOT P 身份验证器。他们还可以设置 MFA 选项并提前注册首选选项。下图列出了用户池配置对初始注册后立即登录尝试的详细影响。

此处说明的逻辑适用于基于 SDK 的应用程序和托管登录登录，但在托管登录中不太明显。当您对 MFA 进行故障排除时，请从用户的结果向后移至促成决策的用户配置文件和用户池配置。



以下列表对应于决策逻辑图中的编号，并详细描述了每个步骤。A

表示身份验证成功且流程已结束。A

表示身份验证失败。

1. 用户在您的登录屏幕上显示其用户名或用户名和密码。如果他们没有出示有效的凭证，他们的登录请求就会被拒绝。
2. 如果他们成功进行用户名密码身份验证，请确定 MFA 是必需的、可选的还是关闭的。如果禁用，则使用正确的用户名和密码即可成功进行身份验证。

a. 如果 MFA 是可选的，请确定用户之前是否设置了 TOTP 身份验证器。如果他们设置了 TOTP，则会提示他们输入 TOTP MFA。如果他们成功回应 MFA 挑战，则表示他们已登录。

b. 确定威胁防护的自适应身份验证功能是否要求用户设置 MFA。如果它尚未分配 MFA，则用户已登录。

3. 如果需要 MFA 或自适应身份验证已分配 MFA，请确定用户是否已将 MFA 因子设置为启用和首选。如果有，请提示使用该因子进行 MFA。如果他们成功回应 MFA 挑战，则表示他们已登录。

4. 如果用户尚未设置 MFA 首选项，请确定用户是否注册了 TOTP 身份验证器。

   a. 如果用户注册了 TOTP 身份验证器，请确定用户池中是否有 TOTP MFA 可用（用户先前设置身份验证器后，可以禁用 TOTP MFA）。

   b. 确定用户池中是否还提供电子邮件或短信 MFA。

   c. 如果电子邮件和短信 MFA 均不可用，请提示用户输入 TOTP MFA。如果他们成功回应 MFA 挑战，则表示他们已登录。

   d. 如果有电子邮件或短信 MFA 可用，请确定用户是否具有相应的`email`或`phone_number`属性。如果是，则任何不是自助账户恢复主要方法且已启用 MFA 的属性都可供他们使用。

   e. 使用包括 TOTP 和可用的 SMS 或电子邮件 MFA 因素在内的`MFAS_CAN_SELECT`选项提示用户`SELECT_MFA_TYPE`提出质疑。

   f. 提示用户输入他们为回应`SELECT_MFA_TYPE`质询而选择的因子。如果他们成功回应 MFA 挑战，则表示他们已登录。

5. 如果用户尚未注册 TOTP 身份验证器，或者他们已注册 TOTP MFA 但当前已禁用，请确定用户是否具有或属性。`email` `phone_number`

6. 如果用户只有电子邮件地址或只有电话号码，请确定该属性是否也是用户池用来发送账户恢复消息以进行密码重置的方法。如果为 true，则他们无法在需要 MFA 的情况下完成登录，并且 Amazon Cognito 会返回错误。要激活该用户的登录，您必须添加非恢复属性或为他们注册 TOTP 身份验证器。

a. 如果他们有可用的非恢复电子邮件地址或电话号码，请确定是否启用了相应的电子邮件或短信 MFA 因子。

b. 如果他们具有非恢复电子邮件地址属性并且启用了电子邮件 MFA，则提示他们提出质疑。`EMAIL_OTP`如果他们成功回应 MFA 挑战，则表示他们已登录。

c. 如果他们具有不可恢复的电话号码属性并且启用了 SMS MFA，请提示他们提出质疑。`SMS_MFA`如果他们成功回应 MFA 挑战，则表示他们已登录。

d. 如果他们没有符合启用电子邮件或短信 MFA 因子的资格，请确定是否启用 TOTP MFA。如果禁用 TOTP MFA，则他们无法在需要 MFA 的情况下完成登录，并且 Amazon Cognito 会返回错误。要激活该用户的登录，您必须添加非恢复属性或为他们注册 TOTP 身份验证器。

> **Note**
>
> 如果用户拥有 TOTP 身份验证器，但 TOTP MFA 已被禁用，则此步骤已被评估为 “否”。

e. 如果启用了 TOTP MFA，请在选项中`MFAS_CAN_SETUP`向`MFA_SETUP`用户提出质疑`SOFTWARE_TOKEN_MFA`。要完成此挑战，您必须单独为用户注册一个 TOTP 身份验证器并使用进行回复。`"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]}"`

f. 用户使用VerifySoftwareToken请求中的会话令牌回应`MFA_SETUP`质询后，提示他们提出`SOFTWARE_TOKEN_MFA`质疑。如果他们成功回应 MFA 挑战，则表示他们已登录。

7. 如果用户同时拥有电子邮件地址和电话号码，请确定哪个属性（如果有）是用于密码重置的帐户恢复消息的主要方法。

a. 如果禁用了自助账户恢复，则任一属性都可用于 MFA。确定是启用电子邮件和短信 MFA 因素中的一个还是两个。

b. 如果两个属性都作为 MFA 因子启用，则使用`MFAS_CAN_SELECT`选项`SMS_MFA`和提示用户`SELECT_MFA_TYPE`提出质疑。`EMAIL_OTP`

c. 提示他们选择应对SELECT_MFA_TYPE挑战的因子。如果他们成功回应 MFA 挑战，则表示他们已登录。

d. 如果只有一个属性是符合条件的 MFA 因子，则提示他们质疑其余因子。如果他们成功回应 MFA 挑战，则表示他们已登录。

这种结果发生在以下场景中。

i. 当它们具有email和phone_number属性时，将启用 SMS 和电子邮件 MFA，而主要的账户恢复方法是通过电子邮件或 SMS 消息。

ii. 当它们具有email和phone_number属性时，仅启用 SMS MFA 或电子邮件 MFA，并且禁用自助账户恢复。

8. 如果用户尚未注册 TOTP 身份验证器，email也没有 nor phone_number 属性，请向他们提出质疑。MFA_SETUP中的列表MFAS_CAN_SETUP包括用户池中所有启用的 MFA 因子，这些因子不是主账户恢复选项。他们可以通过电子邮件或 TOTP MFA 来回应这一挑战。ChallengeResponses要设置 SMS MFA，请单独添加电话号码属性并重新启动身份验证。

对于 TOTP MFA，请使用回复。`"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "SESSION": "[Session ID from VerifySoftwareToken]"}`

对于电子邮件 MFA，请回复。`"ChallengeName": "MFA_SETUP", "ChallengeResponses": {"USERNAME": "[username]", "email": "[user's email address]"}`

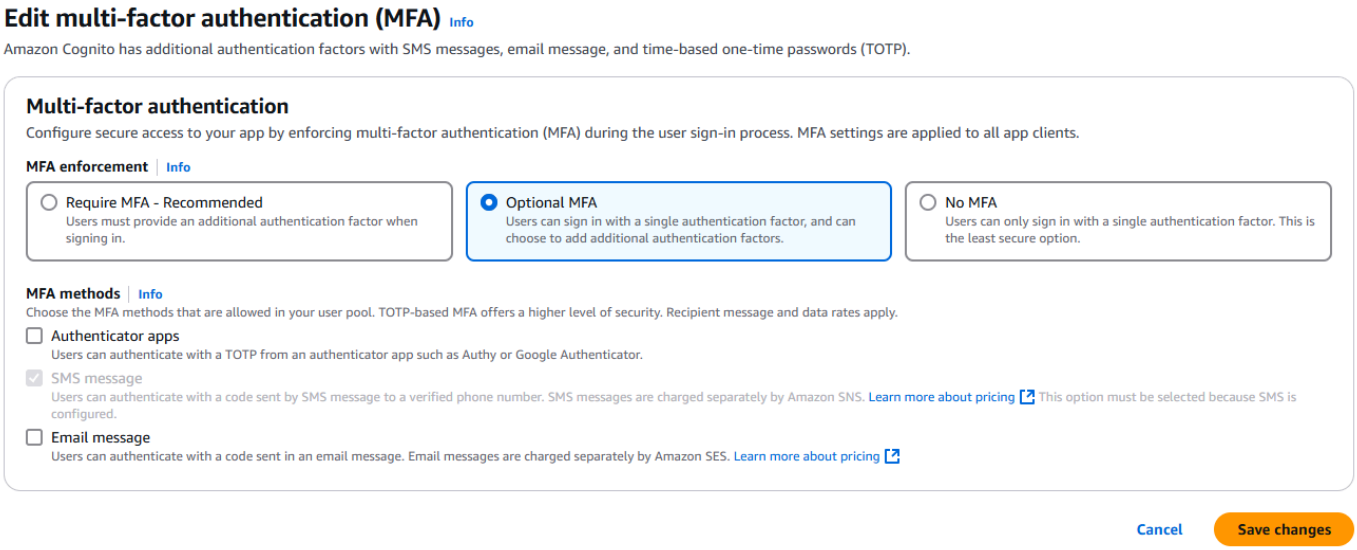
a. 提示他们选择应对SELECT_MFA_TYPE挑战的因子。如果他们成功回应 MFA 挑战，则表示他们已登录。

为多因素身份验证配置用户池

您可以在 Amazon Cognito 控制台中配置 MFA，也可以使用 API 操作和 SD SetUserPoolMfaConfigK 方法进行配置。

在 Amazon Cognito 控制台中配置 MFA

1. 登录 Amazon Cognito 控制台。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “登录” 菜单。找到多重身份验证并选择编辑。
5. 选择您希望用于用户池的MFA 强制执行方法。



   a. 需要 MFA。用户池中的所有用户都必须使用额外的短信、电子邮件或基于时间的一次性密码 (TOTP) 代码作为额外的身份验证因素登录。
   b. 可选 MFA。您可以为用户提供注册额外登录系数的选项，但仍允许尚未配置 MFA 的用户登录。如果您使用自适应身份验证，请选择此选项。有关自适应身份验证的更多信息，请参阅具有威胁防护功能的高级安全性。
   c. 无 MFA。您的用户无法注册其他登录安全要素。
6. 选择您在应用程序中支持的 MFA 方法。您可以将电子邮件消息、短信或 TOTP 生成的身份验证器应用程序设置为第二个因素。
7. 如果使用 SMS 文本消息作为第二安全要素，并且没有配置 IAM 角色与 Amazon Simple Notification Service（Amazon SNS）一起使用 SMS 消息，您可以在控制台中创建一个角色。在用户池的 “身份验证方法” 菜单中，找到 “短信”，然后选择 “编辑”。您还可以使用允许 Amazon Cognito 代表您向用户发送短信的现有角色。有关更多信息，请参阅 IAM 角色。

   如果您使用电子邮件作为第二个因素，但尚未将原始身份配置为与亚马逊简单电子邮件服务 (Amazon SES) Service 一起用于发送电子邮件，请在控制台中创建一个。必须选择 “使用 SES 发送电子邮件” 选项。在用户池的 “身份验证方法” 菜单中，找到 “电子邮件”，然后选

择 “编辑”。从列表中可用的已验证身份中选择一个 FROM 电子邮件地址。例如，如果您选择经过验证的域`example.com`，则还必须在经过验证的域中配置 FROM 发件人姓名`admin-noreply@example.com`。

8. 选择保存更改。

短信和电子邮件消息 MFA

短信和电子邮件 MFA 消息确认用户在登录之前对消息目的地具有访问权限。用户确认他们不仅可以访问密码，还可以访问原始用户的短信或电子邮件收件箱。Amazon Cognito 要求用户提供用户池在成功提供用户名和密码后发送的短代码。

用户向其配置文件添加电子邮件地址或电话号码后，短信和电子邮件 MFA 无需额外配置。Amazon Cognito 可以向未经验证的电子邮件地址和电话号码发送消息。当用户完成第一个 MFA 时，Amazon Cognito 会将其电子邮件地址或电话号码标记为已验证。

当拥有 MFA 的用户在您的应用程序中输入他们的用户名和密码时，MFA 身份验证开始。您的应用程序使用调用InitiateAuth或 AdminInitiateAuthAPI 请求的 SDK 方法提交这些初始参数。API 响应中的`ChallengeParameters` 包括一个 `CODE_DELIVERY_DESTINATION` 值，指明发送授权码的位置。在您的应用程序中，显示一个表单，提示用户查看他们的手机，并包括一个输入框用于输入验证码。在他们输入验证码时，请在质询响应 API 请求中提交验证码以完成登录过程。

拥有 MFA 的用户在托管登录页面中使用用户名和密码登录后，系统会自动提示他们输入 MFA 代码。

用户池使用您的 Amazon Web Services 账户中的 Amazon Simple Notification Service（Amazon SNS）资源发送 MFA 和其他 Amazon Cognito 通知的短信。同样，用户池使用您账户中的 Amazon Simple Email Service（Amazon SES）发送电子邮件消息。这些关联服务在您的邮件传送 Amazon 账单上自行产生费用。在以生产级别的规模发送消息时，还需要满足额外的要求。有关更多信息，请参阅以下链接：

- Amazon Cognito 用户池的短信设置
- 全球短信定价
- Amazon Cognito 用户池的电子邮件设置
- Amazon SES 定价

短信和电子邮件消息 MFA 的注意事项

- 要允许用户使用电子邮件 MFA 登录，您的用户池必须具有以下配置选项：

1. 您的用户池中有 Plus 或 Essentials 功能计划。有关更多信息，请参阅 用户池功能计划。
2. 您的用户池使用您自己的 Amazon SES 资源发送电子邮件消息。有关更多信息，请参阅 Amazon SES 电子邮件配置。

- MFA 验证码在您为应用程序客户端设置的身份验证流程会话持续时间内有效。

  编辑应用程序客户端时，在 Amazon Cognito 控制台的应用程序客户端菜单中设置身份验证流程会话的持续时间。您还可以在 `CreateUserPoolClient` 或 `UpdateUserPoolClient` API 请求中设置身份验证流程会话持续时间。有关更多信息，请参阅 身份验证会话示例。
- 当用户成功提供 Amazon Cognito 发送到未经验证的电话号码或电子邮件地址的短信或电子邮件中的验证码时，Amazon Cognito 会将相应的属性标记为已验证。
- 要让用户自助更改与 MFA 关联的电话号码或电子邮件地址的值，他们必须使用访问令牌登录并授权请求。如果他们无法访问他们当前的电话号码或电子邮件地址，则无法登录。您的团队必须在 AdminUpdateUserAttributesAPI 请求中使用管理员 Amazon 凭据更改这些值。
- 在用户池中配置短信后，即无法禁用将短信作为可用的 MFA 因素。

TOTP 软件令牌 MFA

当您在用户池中设置了 TOTP 软件令牌 MFA 时，您的用户通过用户名和密码登录，然后使用 TOTP 完成身份验证。用户设置并验证用户名和密码后，就可以为 MFA 激活 TOTP 软件令牌。如果您的应用程序使用 Amazon Cognito 托管登录来登录用户，则您的用户将提交他们的用户名和密码，然后在其他登录页面上提交 TOTP 密码。

您可以在 Amazon Cognito 控制台中为用户池激活 TOTP MFA，也可以使用 Amazon Cognito API 操作来激活。在用户池级别，您可以调用SetUserPoolMfaConfig配置 MFA 并启用 TOTP MFA。

> **Note**
>
> 如果您没有为用户池激活 TOTP 软件令牌 MFA，则 Amazon Cognito 无法使用此令牌进行关联或验证用户。在这种情况下，用户会收到 `SoftwareTokenMFANotFoundException` 异常，并带有说明 `Software Token MFA has not been enabled by the userPool`。如果您稍后为用户池停用了软件令牌 MFA，则以前已关联并验证 TOTP 令牌的用户可以继续将其用于 MFA。

为用户配置 TOTP 是一个多步骤过程，在此过程中，用户将收到一个秘密代码，它们通过输入一次性密码来验证该代码。接下来，您可以为用户启用 TOTP MFA，或将 TOTP 设置为用户的首选 MFA 方法。

当您将用户池配置为需要 TOTP MFA 并且您的用户以托管登录方式注册您的应用程序时，Amazon Cognito 会自动执行用户流程。Amazon Cognito 提示您的用户选择 MFA 方法，显示 QR 代码来设置身份验证器应用程序，并验证他们的 MFA 注册。在您允许用户在 SMS 和 TOTP MFA 之间进行选择的用户池中，Amazon Cognito 还为您的用户提供了方法选择。

> **Important**
>
> 当您的 Amazon WAF Web ACL 与用户池相关联，并且您的 Web ACL 中的规则显示了验证码时，这可能会导致托管登录 TOTP 注册中出现不可恢复的错误。要创建具有验证码操作且不影响托管登录 TOTP 的规则，请参阅。为托管登录 TOTP MFA 配置您的 Amazon WAF 网页 ACL有关 Amazon WAF 网络 ACLs 和 Amazon Cognito 的更多信息，请参阅。将 Amazon WAF Web ACL 与用户池关联

要使用软件开发工具包和 Amazon Co gnito 用户池 API 在自定义用户界面 Amazon 中实现 TOTP MFA，请参阅。为用户配置 TOTP MFA

要向用户池添加 MFA，请参阅 向用户池添加 MFA。

TOTP MFA 注意事项和限制

1. Amazon Cognito 通过可生成 TOTP 代码的身份验证器应用程序支持软件令牌 MFA。Amazon Cognito 不支持基于硬件的 MFA。
2. 当您的用户池要求尚未进行配置的用户提供 TOTP 时，您的用户将收到一个一次性访问令牌，应用程序可使用该令牌为用户激活 TOTP MFA。在用户注册了额外的 TOTP 登录安全要素之前，后续的登录尝试将失败。
   - 使用 `SignUp` API 操作或通过托管登录在您的用户池中注册的用户将在用户完成注册后收到一次性令牌。
   - 在您创建用户并且该用户设置了初始密码后，Amazon Cognito 会通过托管登录向该用户发放一次性令牌。如果您为用户设置了永久密码，则 Amazon Cognito 会在用户首次登录时颁发一次性令牌。
   - Amazon Cognito 不会向使用或 API 操作登录的管理员创建的用户发放一次性令牌。InitiateAuthAdminInitiateAuth在您的用户成功完成质询来设置初始密码后，或者如果您为用户设置永久密码，Amazon Cognito 会立即向用户提出质询来设置 MFA。
3. 如果需要 MFA 的用户池中的用户已收到一次性访问令牌但尚未设置 TOTP MFA，则该用户在设置 MFA 之前无法使用托管登录进行登录。您可以代替访问令牌，而是在请求中使用`MFA_SETUP`质询InitiateAuth或AssociateSoftwareToken请求AdminInitiateAuth中的`session`响应值。

4. 如果您的用户已设置 TOTP，则他们可以将其用于 MFA，即使您以后停用用户池的 TOTP。
5. Amazon Cognito TOTPs 仅接受使用 HMAC 哈希函数生成代码的身份验证器应用程序。SHA1 使用 SHA-256 哈希生成的代码会返回 `Code mismatch` 错误。

为用户配置 TOTP MFA

当用户首次登录时，您的应用程序使用他们的一次性访问令牌生成 TOTP 私钥，并以文本或二维码格式将其呈现给用户。您的用户配置其身份验证器应用程序并为后续登录尝试提供 TOTP。您的应用程序或托管登录会在 MFA 质询响应中向 Amazon Cognito 显示 TOTP。

在某些情况下，托管登录会提示新用户设置 TOTP 身份验证器。有关更多信息，请参阅。用户运行时的 MFA 逻辑详情

主题

- 关联 TOTP 软件令牌
- 验证 TOTP 令牌
- 使用 TOTP MFA 登录
- 删除 TOTP 令牌

关联 TOTP 软件令牌

要关联 TOTP 令牌，请向用户发送一个必须使用一次性密码来验证的秘密代码。关联令牌需要执行三个步骤。

1. 当您的用户选择 TOTP 软件令牌 MFA 时，AssociateSoftwareToken调用返回为用户帐户生成的唯一共享密钥代码。您可以使用访问令牌或会话字符串进行授权 AssociateSoftwareToken 。
2. 您的应用程序向用户呈现私钥或您从私钥生成的二维码。用户必须将密钥输入到一个生成 TOTP 的应用程序（如 Google Authenticator）中。您可以使用 libqrencode 生成二维码。
3. 您的用户用身份验证器应用程序（例如 Google Authenticator）输入密钥或扫描二维码，该应用程序开始生成代码。

验证 TOTP 令牌

接下来验证 TOTP 令牌。要求您的用户提供示例代码并将其提供给 Amazon Cognito 服务，以确认用户成功生成 TOTP 代码，如下所示。

1. 您的应用程序会提示用户输入代码，以证明他们已正确设置了身份验证器应用程序。

2. 用户的身份验证器应用程序显示一个临时密码。身份验证器应用程序根据您提供给用户的密钥生成密码。
3. 用户输入他们的临时密码。您的应用程序在 `VerifySoftwareToken` API 请求中将临时密码传递给 Amazon Cognito。
4. Amazon Cognito 保留与用户关联的密钥，并生成 TOTP 并将其与用户提供的 TOTP 进行比较。如果匹配，`VerifySoftwareToken` 返回 `SUCCESS` 响应。
5. Amazon Cognito 将 TOTP 要素与用户关联起来。
6. 如果 `VerifySoftwareToken` 操作返回 `ERROR` 响应，请确保用户的时钟正确且没有超过最大重试次数。Amazon Cognito 接受在尝试前后 30 秒之内的 TOTP 令牌，以容许轻微的时钟偏差。解决问题后，请重试该 VerifySoftwareToken 操作。

使用 TOTP MFA 登录

此时，您的用户可使用基于时间的一次性密码登录。过程如下所述。

1. 用户将输入其用户名和密码来登录客户端应用程序。
2. TOTP MFA 质询被调用，您的应用程序提示用户输入临时密码。
3. 您的用户从关联的 TOTP 生成应用程序获取临时密码。
4. 您的用户在您的客户端应用程序中输入 TOTP 代码。您的应用程序通知 Amazon Cognito 服务以验证该代码。对于每次登录，都RespondToAuthChallenge应调用以获取对新 TOTP 身份验证质询的响应。
5. 如果令牌通过 Amazon Cognito 验证，则登录成功，您的用户可以继续完成身份验证流程。

删除 TOTP 令牌

最后，您的应用程序应允许您的用户停用他们的 TOTP 配置。当前，您无法删除用户的 TOTP 软件令牌。要替换用户的软件令牌，请关联并验证新的软件令牌。要为用户停用 TOTP MFA，请致电SetUserMFAPreference将您的用户修改为不使用 MFA 或仅使用 SMS MFA。

1. 在您的应用程序中为想要重置 MFA 的用户创建界面。在此界面中提示用户输入密码。
2. 如果 Amazon Cognito 返回 TOTP MFA 质询，请将用户的 MFA 偏好设置更新为。SetUserMFAPreference
3. 在您的应用程序中，告知您的用户他们已停用 MFA 并提示他们重新登录。

为托管登录 TOTP MFA 配置您的 Amazon WAF 网页 ACL

当您的 Amazon WAF Web ACL 与用户池相关联，并且您的 Web ACL 中的规则显示了验证码时，这可能会导致托管登录和管理登录 TOTP 注册中出现不可恢复的错误。 Amazon WAF 验证码规则仅对托管登录和经典托管用户界面中的 TOTP MFA 产生这种影响。SMS MFA 不受影响。

当 CAPTCHA 规则不允许用户完成 TOTP MFA 设置时，Amazon Cognito 会显示以下错误。

由于 WAF captcha，不允许请求。

当你的用户池在后台发出的 VerifySoftwareTokenAPI 请求时 Amazon WAF 提示输入验证码时，就会出现此错误。AssociateSoftwareToken要创建具有验证码操作且不影响托管登录页面中的 TOTP 的规则，请在规则中`VerifySoftwareToken`从 CAPTCHA 操作中排除`AssociateSoftwareToken`和的`x-amzn-cognito-operation-name`标题值。

以下屏幕截图显示了一个示例 Amazon WAF 规则，该规则将 CAPTCHA 操作应用于`x-amzn-cognito-operation-name`标头值不为或的所有请求。`AssociateSoftwareToken` `VerifySoftwareToken`

If a request matches all the statements (AND)

NOT Statement 1

Field to match

Single header (x-amzn-cognito-operation-name)

Positional constraint

Exactly matches string

Search string

AssociateSoftwareToken

Text transformations

- None (Priority 0)

AND

NOT Statement 2

Field to match

Single header (x-amzn-cognito-operation-name)

Positional constraint

Exactly matches string

Search string

VerifySoftwareToken

Text transformations

- None (Priority 0)

Then

Action

The action to take when a web request matches the rule statement.

有关 Amazon WAF 网络 ACLs 和 Amazon Cognito 的更多信息，请参阅。将 Amazon WAF Web ACL 与用户池关联

具有威胁防护功能的高级安全性

创建用户池后，您可以在 Amazon Cognito 控制台的导航菜单中访问威胁防护。您可以开启威胁防护功能并自定义为应对不同风险而采取的操作。或者，您可以使用审计模式收集与检测到的风险相关的指标，而无需应用任何安全缓解措施。在审计模式下，威胁防护会向 Amazon 发布指标 CloudWatch。在 Amazon Cognito 生成第一个事件之后，您可以看到指标。请参阅查看威胁防护指标。

威胁防护（以前称为高级安全功能）是一组针对用户池中不想要的活动的监控工具，以及用于自动关闭潜在恶意活动的配置工具。威胁防护为标准和自定义身份验证操作提供了不同的配置选项。例如，在设置了额外安全因素时，您可能想向使用可疑自定义身份验证登录的用户发送通知，而阻止使用基本的用户名和密码身份验证且处于相同风险级别的用户。

Plus 功能计划中提供了威胁防护。有关更多信息，请参阅 用户池功能计划。

以下用户池选项是威胁防护的组件。

已泄露的凭证

用户将密码重复用于多个用户账户。Amazon Cognito 的已泄露凭证功能可编译公开泄露的用户名和密码数据，并将用户的凭证与泄露的凭证列表进行比较。已泄露凭证的检测还检查常猜测的密码。您可以在用户池的 username-and-password标准身份验证流程中检查凭据是否被泄露。Amazon Cognito 不会在安全远程密码（SRP）或自定义身份验证流程中检测凭证是否泄露。

您可以选择用于提示检查已泄露凭证的用户操作，以及您希望 Amazon Cognito 采取的应对措施。对于登录、注册和密码更改事件，Amazon Cognito 可以禁止登录或允许登录。在这两种情况下，Amazon Cognito 都会生成用户活动日志，您可以在其中找到有关该事件的更多信息。

了解更多

使用已泄露的凭证检测

自适应身份验证

Amazon Cognito 可以查看来自用户登录请求的位置和设备信息，并应用自动响应来保护用户池中的用户账户免受可疑活动的侵害。您可以监控用户活动，并自动响应在用户名密码和 SRP 以及自定义身份验证中检测到的风险级别。

当您激活威胁防护时，Amazon Cognito 会为用户活动分配风险评分。您可以为可疑活动指定自动响应：您可以需要 MFA、禁止登录，或者只记录活动详细信息和风险评分。您还可以自动发送电子邮件，将可疑活动通知用户，以便他们可以重置密码或采取其他自助操作。

了解更多

使用自适应身份验证

IP 地址允许列表和拒绝列表

在全功能模式下使用 Amazon Cognito 威胁防护，您可以创建 IP 地址 “始终阻止” 和 “始终允许例外”。对于来自始终阻止例外列表中 IP 地址的会话，自适应身份验证不会向其分配风险级别，该会话也无法登录您的用户池。

有关 IP 地址许可名单和屏蔽名单的注意事项

- 必须以 CIDR 格式表示 “始终阻止” 和 “始终允许”`192.0.2.0/24`，例如，24 位掩码或`192.0.2.252/32`单个 IP 地址。
- IP 地址在 “始终阻止” IP 范围内的设备无法使用基于 SDK 的或托管登录应用程序进行注册或登录，但它们可以使用第三方登录。 IdPs
- “始终允许” 和 “始终阻止” 列表不会影响令牌刷新。
- Amazon Cognito 不会将自适应身份验证 MFA 规则应用于 “始终允许” IP 范围内的设备，但会应用受损凭证规则。

日志导出

威胁防护会将用户身份验证请求的详细信息记录到您的用户池中。这些日志包含威胁评估、用户信息以及位置和设备等会话元数据。您可以为这些日志创建外部归档以进行保留和分析。Amazon Cognito 用户池将威胁防护日志导出到亚马逊 S3、 CloudWatch 日志和亚马逊数据 Firehose。有关更多信息，请参阅 查看和导出用户事件历史记录。

了解更多

导出威胁防护用户活动日志

主题

- 威胁防护的注意事项和限制
- 在用户池中开启威胁防护
- 威胁防护强制执行概念
- 标准身份验证和自定义身份验证的威胁防护

- 威胁防护先决条件
- 设置威胁防护
- 使用已泄露的凭证检测
- 使用自适应身份验证
- 在应用程序中收集威胁防护的数据

威胁防护的注意事项和限制

不同身份验证流程的威胁防护选项有所不同

Amazon Cognito 通过身份验证流程 `USER_PASSWORD_AUTH` 和 `ADMIN_USER_PASSWORD_AUTH` 来支持自适应身份验证和凭证泄露检测。您可以仅为 `USER_SRP_AUTH` 启用自适应身份验证。不能将威胁防护与联合身份验证登录一起使用。

始终屏蔽会增加 IPs 请求配额

在您的用户池中阻止来自始终阻止例外列表中 IP 地址的请求，可以帮助您的用户池保持在请求速率配额以内。

威胁防护不会应用速率限制

一些恶意流量具有请求量大的特征，例如分布式拒绝服务 (DDoS) 攻击。Amazon Cognito 对传入流量应用的风险评分是基于单个请求的，并不考虑请求数。在大批量事件中，即便某些请求与大规模攻击无关，它们仍然可能因为应用层的原因而被赋予风险评分，并触发自动化响应。要对用户池中的容量攻击实施防御，请添加 Amazon WAF Web。 ACLs有关更多信息，请参阅 将 Amazon WAF Web ACL 与用户池关联。

威胁防护不影响 M2M 请求

客户凭证授予的目的是在与用户帐户无关的情况下进行 machine-to-machine（M2M）授权。威胁防护仅监控用户池中的用户帐户和密码。要在 M2M 活动中实现安全功能，请考虑 Amazon WAF 用于监控请求率和内容的功能。有关更多信息，请参阅 将 Amazon WAF Web ACL 与用户池关联。

在用户池中开启威胁防护

Amazon Cognito user pools console

为用户池激活威胁防护

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。

2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 如果您还没有，请从 “设置” 菜单中激活 Plus 功能计划。
5. 选择 “威胁防护” 菜单，然后选择 “激活”。
6. 选择保存更改。

API

在CreateUserPool或 UpdateUserPoolAPI 请求中将您的功能计划设置为 Plus。以下部分示例请求正文将威胁防护设置为全功能模式。有关完整的示例请求，请参阅示例。

```
"UserPoolAddOns": {
      "AdvancedSecurityMode": "ENFORCED"
   }
```

威胁防护是一个总括性术语，指的是一组特征，用于监控用户操作以发现账户盗用的迹象，并自动采取措施来保护受影响的用户账户。当用户使用标准和自定义身份验证流程登录时，您可以对用户应用威胁防护设置。

威胁防护会生成日志，详细说明用户的登录、注销和其他活动。您可以将这些日志导出到第三方系统。有关更多信息，请参阅 查看和导出用户事件历史记录。

威胁防护强制执行概念

威胁防护从仅限审计模式开始，在此模式下，用户池会监控用户活动、分配风险级别和生成日志。妥善的做法是，在启用完全功能模式之前，在仅限审计模式下运行两周或更长时间。完全功能模式包括一组针对检测到的风险活动和密码泄露的自动响应措施。在仅限审计模式下，您可以监控 Amazon Cognito 正在执行的威胁评估。您还可以提供反馈，帮助训练该特征以改进误报和漏报。

您可以在用户池级别配置威胁防护强制执行措施，以覆盖用户池中的所有应用程序客户端，也可以在单个应用程序客户端级别配置威胁防护。应用程序客户端威胁防护配置会覆盖用户池配置。要为应用程序客户端配置威胁防护，请在 Amazon Cognito 控制台的用户池的应用程序客户端菜单中导航到应用程序客户端设置。在这里，您可以使用客户端级设置并配置应用程序客户端专有的强制执行。

此外，您可以分别为标准和自定义身份验证类型配置威胁防护。

标准身份验证和自定义身份验证的威胁防护

配置威胁防护的方式取决于您在用户池和应用程序客户端中进行的身份验证类型。以下每种类型的身份验证都可以有自己的强制执行模式和自动响应。

标准身份验证

标准身份验证是用户登录、注销和密码管理，包括用户名密码流和托管登录。当操作使用托管登录或使用以下 API `AuthFlow` 参数登录时，Amazon Cognito 威胁防护会监控操作是否存在风险指标：

InitiateAuth

`USER_PASSWORD_AUTH`、`USER_SRP_AUTH`。泄露的凭证特征无法在 `USER_SRP_AUTH` 登录时访问密码，并且不会监控或对这个流程中的事件采取行动。

AdminInitiateAuth

`ADMIN_USER_PASSWORD_AUTH`、`USER_SRP_AUTH`。泄露的凭证特征无法在 `USER_SRP_AUTH` 登录时访问密码，并且不会监控或对这个流程中的事件采取行动。

您可以将标准身份验证的强制执行模式设置为仅限审计或完全功能。要禁用标准身份验证的威胁监控，请将威胁防护设置为不强制执行。

自定义身份验证

自定义身份验证是使用自定义质询 Lambda 触发器的用户登录。您无法在托管登录中进行自定义身份验证。当用户使用 `InitiateAuth` 和 `AdminInitiateAuth` 的 API `AuthFlow` 参数 `CUSTOM_AUTH` 登录时，Amazon Cognito 威胁防护会监控操作的风险指标。

您可以将自定义身份验证的强制执行模式设置为仅限审计、完全功能或没有强制执行。“不强制执行” 选项可在不影响其他威胁防护功能的情况下禁用自定义身份验证的威胁监控。

威胁防护先决条件

在开始之前，您需要：

- 用户池和应用程序客户端。有关更多信息，请参阅 用户池入门。
- 在 Amazon Cognito 控制台中将多重身份验证（MFA）设置为可选,以使用基于风险的自适应身份验证功能。有关更多信息，请参阅 向用户池添加 MFA。
- 如果您使用电子邮件通知，请转到 Amazon SES 控制台配置并验证要用于通知电子邮件的电子邮件地址或域。有关 Amazon SES 的更多信息，请参阅在 Amazon SES 中验证身份。

设置威胁防护

按照以下说明设置用户池威胁防护。

> **Note**
>
> 要在 Amazon Cognito 用户池控制台中为应用程序客户端设置不同的威胁防护配置，请从应用程序客户端菜单中选择应用程序客户端，然后选择使用客户端级设置。

Amazon Web Services Management Console

为用户池配置威胁防护

1. 转到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “威胁防护” 菜单，然后选择 “激活”。
5. 选择要配置的威胁防护方法：标准身份验证和自定义身份验证。您可以为自定义身份验证和标准身份验证设置不同的强制执行模式，但在完全功能模式下，它们共享自动响应的配置。
6. 选择编辑。
7. 选择强制执行模式。要立即开始对检测到的风险作出响应，请选择完全功能并配置针对已泄露凭证和自适应身份验证的自动响应。要在用户级日志中收集信息 CloudWatch，请选择 “仅审计”。

   我们建议您在启用操作之前，将威胁防护保持在审核模式下两周。在此期间，Amazon Cognito 可以了解应用程序用户的使用模式，并且您可以提供事件反馈以调整响应。
8. 如果您已选择仅审计，请选择保存更改。如果您已选择完整功能：

   a. 选择是否要进行自定义操作或使用 Cognito 默认设置响应可疑已泄露的凭证。Cognito 默认设置：

      i. 检测登录、注册和密码更改中的已泄露的凭证。

      ii. 使用禁止登录操作响应已泄露的凭证。

   b. 如果您为已泄露的凭证选择了自定义操作，请选择 Amazon Cognito 将用于事件检测的用户池操作，以及您希望 Amazon Cognito 执行的 已泄露凭证的响应。您可以使用可疑的已泄露凭证进行禁止登录或允许登录。

c. 在自适应身份验证下，选择如何响应恶意登录尝试。选择是否要进行自定义操作或使用 Cognito 默认设置响应可疑恶意活动。当您选择 Cognito 默认设置时，Amazon Cognito 会阻止所有风险级别的登录，并且不会通知用户。

d. 如果您针对自适应身份验证已选择自定义操作，请根据严重性级别选择 Amazon Cognito 对检测到的风险执行的 自动风险响应操作。当您针对风险级别分配响应时，您无法为较高风险级别分配限制性较小的响应。您可以为风险级别分配以下响应：

   i. 允许登录 – 不采取任何预防性操作。

   ii. 可选 MFA – 如果用户配置了 MFA，Amazon Cognito 将始终要求用户在登录时提供其它 SMS 或基于时间的一次性密码（TOTP）因素。如果用户没有配置 MFA，他们可以继续正常登录。

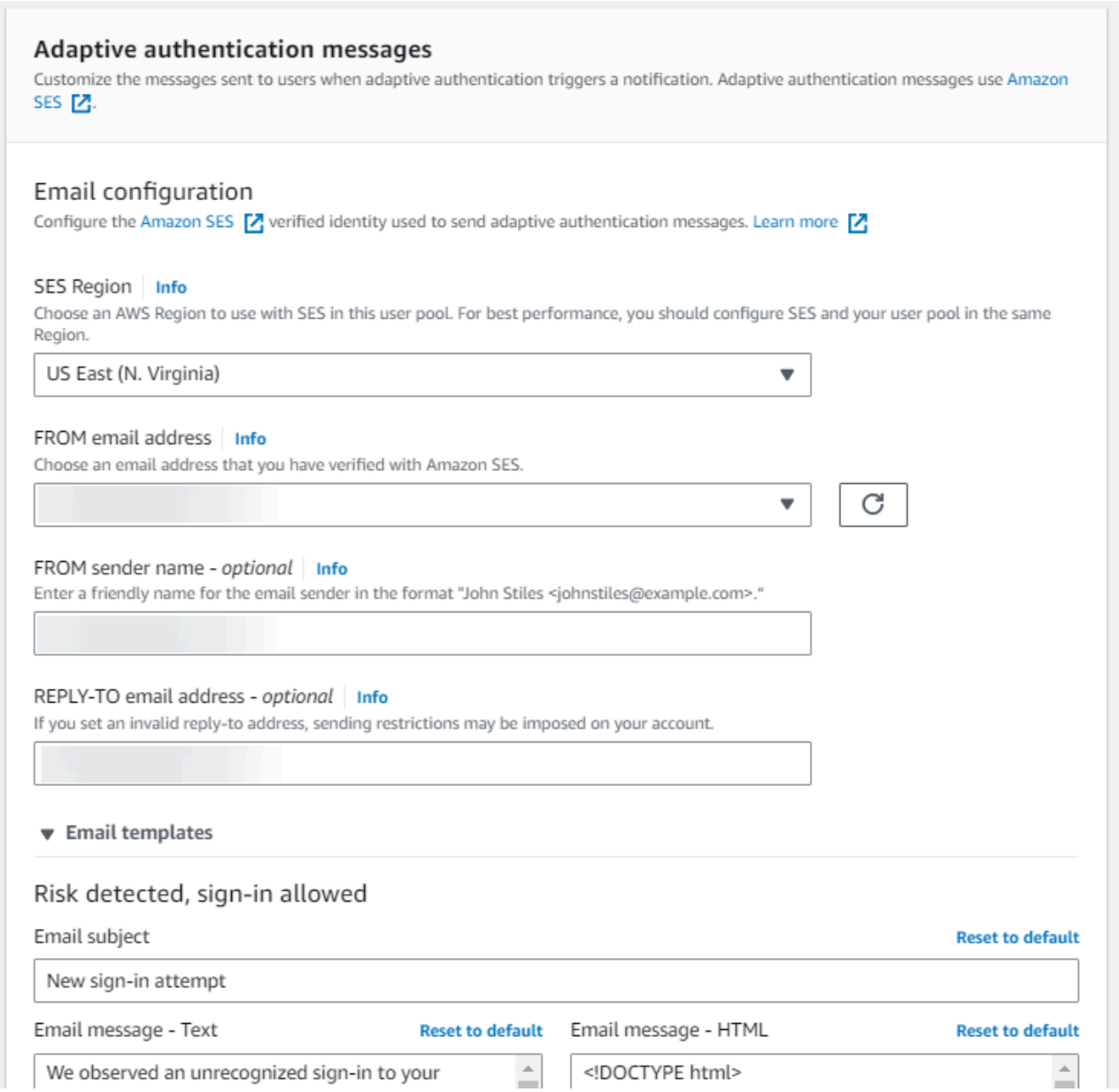
   iii. 需要 MFA – 如果用户配置了 MFA，Amazon Cognito 将始终要求用户在登录时提供其它短信或 TOTP 因素。如果用户没有配置 MFA，Amazon Cognito 将提示他们设置 MFA。在您自动要求用户使用 MFA 之前，请在应用程序中配置一种机制来捕获 SMS MFA 的电话号码，或为 TOTP MFA 注册身份验证器应用程序。

   iv. 禁止登录 – 阻止用户登录。

   v. 通知用户 – 向用户发送电子邮件，其中包含有关 Amazon Cognito 检测到的风险以及您所采取的响应的信息。您可以为发送的消息自定义电子邮件消息模板。

9. 如果您在上一步骤中选择了通知用户，您可以自定义电子邮件发送设置和电子邮件消息模板以进行自适应身份验证。

   a. 在邮件配置下，选择您希望与自适应身份验证一起使用的 SES 区域、FROM 电子邮件地址、FROM 发件人名称和 REPLY-TO 电子邮件地址。有关将用户池电子邮件消息与 Amazon Simple Email Service 集成的更多信息，请参阅 Amazon Cognito 用户池的电子邮件设置。



b. 展开电子邮件模板以自定义包含 HTML 和纯文本版本电子邮件消息的自适应身份验证通知。要了解有关电子邮件消息模板的更多信息，请参阅消息模板。

10. 扩展 IP 地址例外以创建始终允许或始终阻止的列表 IPv4 或 IPv6 地址范围，无论威胁防护风险评估如何，这些地址范围都将始终被允许或阻止。用 CIDR 表示法指定 IP 地址范围（例如，192.168.100.0/24）。

11. 选择保存更改。

API (user pool)

要为用户池设置威胁防护配置，请发送包含参数但不包含`UserPoolId`参数的 SetRiskConfigurationAPI 请求。`ClientId`以下是用户池的一个示例请求正文。此风险配置会根据风险的严重程度执行一系列不断升级的操作，并通知所有风险级别的用户。它对注册操作应用已泄露凭证阻止。

要强制执行此配置，您必须在单独的请求CreateUserPool或 UpdateUserPoolAPI 请求`ENFORCED`中`AdvancedSecurityMode`将设置为。有关占位符模板的更多信息（如本例中的`{username}`），请参阅 配置验证和邀请消息。

```
{
   "AccountTakeoverRiskConfiguration": {
      "Actions": {
         "HighAction": {
            "EventAction": "MFA_REQUIRED",
            "Notify": true
         },
         "LowAction": {
            "EventAction": "NO_ACTION",
            "Notify": true
         },
         "MediumAction": {
            "EventAction": "MFA_IF_CONFIGURED",
            "Notify": true
         }
      },
      "NotifyConfiguration": {
         "BlockEmail": {
            "Subject": "You have been blocked for suspicious activity",
            "TextBody": "We blocked {username} at {login-time} from {ip-address}."
         },
         "From": "admin@example.com",
         "MfaEmail": {
            "Subject": "Suspicious activity detected, MFA required",
            "TextBody": "Unexpected sign-in from {username} on device {device-name}.
 You must use MFA."
         },
         "NoActionEmail": {
            "Subject": "Suspicious activity detected, secure your user account",
            "TextBody": "We noticed suspicious sign-in activity by {username} from
 {city}, {country} at {login-time}. If this was not you, reset your password."
         },
```

```
         "ReplyTo": "admin@example.com",
         "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
      }
   },
   "CompromisedCredentialsRiskConfiguration": {
      "Actions": {
         "EventAction": "BLOCK"
      },
      "EventFilter": [ "SIGN_UP" ]
   },
   "RiskExceptionConfiguration": {
      "BlockedIPRangeList": [ "192.0.2.0/24","198.51.100.0/24" ],
      "SkippedIPRangeList": [ "203.0.113.0/24" ]
   },
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

API (app client)

要为应用程序客户端设置威胁防护配置，请发送包含`UserPoolId`参数和参数的 SetRiskConfigurationAPI 请求。`ClientId`以下是应用程序客户端的一个示例请求正文。此风险配置比用户池配置更为严格，会阻止高风险条目。该配置还将已泄露的凭证阻止应用于注册、登录和密码重置操作。

要强制执行此配置，您必须在单独的请求CreateUserPool或 UpdateUserPoolAPI 请求`ENFORCED`中`AdvancedSecurityMode`将设置为。有关占位符模板的更多信息（如本例中的`{username}`），请参阅 配置验证和邀请消息。

```
{
   "AccountTakeoverRiskConfiguration": {
      "Actions": {
         "HighAction": {
            "EventAction": "BLOCK",
            "Notify": true
         },
         "LowAction": {
            "EventAction": "NO_ACTION",
            "Notify": true
         },
         "MediumAction": {
            "EventAction": "MFA_REQUIRED",
            "Notify": true
```

```
         }
      },
      "NotifyConfiguration": {
         "BlockEmail": {
            "Subject": "You have been blocked for suspicious activity",
            "TextBody": "We blocked {username} at {login-time} from {ip-address}."
         },
         "From": "admin@example.com",
         "MfaEmail": {
            "Subject": "Suspicious activity detected, MFA required",
            "TextBody": "Unexpected sign-in from {username} on device {device-name}.
 You must use MFA."
         },
         "NoActionEmail": {
            "Subject": "Suspicious activity detected, secure your user account",
            "TextBody": "We noticed suspicious sign-in activity by {username} from
 {city}, {country} at {login-time}. If this was not you, reset your password."
         },
         "ReplyTo": "admin@example.com",
         "SourceArn": "arn:aws:ses:us-west-2:123456789012:identity/
admin@example.com"
      }
   },
   "ClientId": "1example23456789",
   "CompromisedCredentialsRiskConfiguration": {
      "Actions": {
         "EventAction": "BLOCK"
      },
      "EventFilter": [ "SIGN_UP", "SIGN_IN", "PASSWORD_CHANGE" ]
   },
   "RiskExceptionConfiguration": {
      "BlockedIPRangeList": [ "192.0.2.1/32","192.0.2.2/32" ],
      "SkippedIPRangeList": [ "192.0.2.3/32","192.0.2.4/32" ]
   },
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

使用已泄露的凭证检测

Amazon Cognito 可以检测用户的用户名和密码是否已在别处遭盗用。这种情况可能出现在用户在多个站点重复使用凭证时或它们使用不安全的密码时。Amazon Cognito 会检查使用用户名和密码、托管登

录和亚马逊 Cognito API 登录的本地用户。本地用户仅存在于您的用户池目录中，无需通过外部 IdP 进行联合身份验证。

在 Amazon Cognito 控制台的威胁防护菜单中，您可以配置已泄露的凭证。配置事件检测以选择要监控的用户事件，以查看是否有已泄露的凭证。配置已泄露凭证的响应，以选择在检测到已泄露的凭证时是允许还是阻止用户。Amazon Cognito 可以在登录、注册和密码更改期间检查是否有已泄露的凭证。

选择 “允许登录” 后，您可以查看 Amazon CloudWatch Logs 以监控 Amazon Cognito 对用户事件所做的评估。有关更多信息，请参阅 查看威胁防护指标。当您选择禁止登录时，Amazon Cognito 会阻止使用已泄露的凭证的用户登录。当 Amazon Cognito 阻止用户登录时，它将用户的 `UserStatus` 设置为 `RESET_REQUIRED`。具有 `RESET_REQUIRED` 状态的用户必须先更改密码，然后才能再次登录。

> **Note**
>
> 当前，对于采用安全远程密码（SRP）流程的登录操作，Amazon Cognito 不会检查是否有已泄露的凭证。SRP 在登录期间发送经过哈希处理的密码证明。Amazon Cognito 无法在内部访问密码，因此，它只能评估您的客户端以纯文本形式传递给它的密码。
> Amazon Cognito 会检查使用带流AdminInitiateAuth的 API 和带`ADMIN_USER_PASSWORD_AUTH`流的 InitiateAuthAPI 的登录凭证是否遭到`USER_PASSWORD_AUTH`泄露。

要向用户池添加已泄露的凭证保护，请参阅 具有威胁防护功能的高级安全性。

使用自适应身份验证

借助自适应身份验证，可以将用户池配置为阻止可疑登录，或为响应增加的风险级别添加第二安全要素身份验证。对于每次登录尝试，Amazon Cognito 都会生成一个风险分数来表示登录请求来自遭盗用源的可能性。此风险评分基于您的应用程序提供的设备和用户因素，以及 Amazon Cognito 从请求中得出的其他因素。促成 Amazon Cognito 进行风险评估的一些因素包括 IP 地址、用户代理以及与其他登录尝试的地理距离。当 Amazon Cognito 检测到用户会话中存在风险且用户尚未选择多重身份验证（MFA）方法时，自适应身份验证可以为用户池中的用户开启或要求 MFA。当您为用户激活 MFA 时，无论您如何配置自适应身份验证，他们都会收到在身份验证期间提供或设置第二因素的质询。从用户的角度来看，您的应用程序可以帮助他们设置 MFA，也可以选择让 Amazon Cognito 阻止他们再次登录，直到他们配置了附加因素。

Amazon Cognito 向亚马逊发布了有关登录尝试、其风险等级和挑战失败的指标。 CloudWatch有关更多信息，请参阅 查看威胁防护指标。

要向用户池添加自适应身份验证，请参阅 具有威胁防护功能的高级安全性。

主题

- 自适应身份验证概览
- 将用户设备和会话数据添加到 API 请求
- 查看和导出用户事件历史记录
- 提供事件反馈
- 发送通知消息

自适应身份验证概览

从 Amazon Cognito 控制台的威胁防护菜单中，您可以选择自适应身份验证的设置，包括在不同风险级别下要采取的操作以及自定义向用户发送的通知消息。您可以为所有应用程序客户端分配全局威胁防护配置，但要将客户端级别的配置应用于各个应用程序客户端。

Amazon Cognito 自适应身份验证为每个用户会话分配以下风险级别之一：高、中、低或无风险。

将强制执行方法从仅限审计更改为完整功能时，请仔细斟酌您的选项。您对风险等级应用的自动响应会影响 Amazon Cognito 为具有相同特征的后续用户会话分配的风险等级。例如，在您选择不采取任何操作或允许之后，对于 Amazon Cognito 最初评估为高风险的用户会话，Amazon Cognito 会认为类似会话的风险较低。

对于每个风险级别，您可以选择以下选项：

选项	操作
允许	用户无需额外安全要素即可登录。
可选 MFA	已配置第二安全要素的用户需要完成第二安全要素质询才能登录。用于 SMS 的电话号码和 TOTP 软件令牌是可供使用的第二个安全要素。未配置第二个因素的用户只能使用一组凭证登录。
需要 MFA	已配置第二安全要素的用户需要完成第二安全要素质询才能登录。Amazon Cognito 阻止未配置第二个安全要素的用户登录。

选项	操作
阻止	Amazon Cognito 阻止指定风险级别下的所有登录尝试。

> **Note**
> 您无需验证手机号码即可将其用于 SMS 来作为第二个身份验证要素。

将用户设备和会话数据添加到 API 请求

当您使用 Amazon Cognito 威胁防护 API 进行注册、登录和重置密码时，您可以收集有关用户会话的信息并将其传递给 Amazon Cognito 威胁防护。此信息包括用户的 IP 地址和唯一的设备标识符。

您可能在用户和 Amazon Cognito 之间有中间网络设备，例如代理服务或应用程序服务器。您可以收集用户的上下文数据并将其传递给 Amazon Cognito，以便自适应身份验证根据用户端点（而不是服务器或代理）的特征来计算风险。如果您的客户端应用程序直接调用 Amazon Cognito API 操作，自适应身份验证会自动记录源 IP 地址。但是，它不会记录其他设备信息（例如 `user-agent`），除非您也收集设备指纹。

使用 Amazon Cognito 上下文数据收集库生成这些数据，然后使用和参数将其提交给 Amazon Cognito 威胁防护。ContextDataUserContextData上下文数据收集库包含在 Amazon SDKs。有关更多信息，请参阅 将 Amazon Cognito 身份验证和授权与 Web 和移动应用程序集成。`ContextData`如果您有 Plus 功能计划，则可以提交。有关更多信息，请参阅 设置威胁防护。

当您从应用程序服务器调用以下经过 Amazon Cognito 身份验证的 API 操作时，请在 `ContextData` 参数中传递用户设备的 IP。此外，请传递服务器名称、服务器路径和编码的设备指纹数据。

- AdminInitiateAuth
- AdminRespondToAuthChallenge

当您调用 Amazon Cognito 未经身份验证的 API 操作时，您可以提交到亚马逊 `UserContextData` Cognito 威胁防护。此数据在 `EncodedData` 参数中包括设备指纹。如果您符合以下条件，也可以在 `UserContextData` 中提交 `IpAddress` 参数：

- 您的用户池已加入 Plus 功能套餐。有关更多信息，请参阅 用户池功能计划。

- 您的应用程序客户端具有客户端密钥。有关更多信息，请参阅 特定于应用程序的应用程序客户端设置。
- 您已在应用程序客户端中激活 接受其他用户上下文数据。有关更多信息，请参阅 接受额外的用户上下文数据（Amazon Web Services Management Console）。

您的应用程序可以在以下 Amazon Cognito 未经验证的 API 操作中，使用编码的设备指纹数据和用户设备的 IP 地址填充 `UserContextData` 参数。

- InitiateAuth
- RespondToAuthChallenge
- SignUp
- ConfirmSignUp
- ForgotPassword
- ConfirmForgotPassword
- ResendConfirmationCode

接受额外的用户上下文数据（Amazon Web Services Management Console）

激活接受其他用户上下文数据功能后，用户池在 `UserContextData` 参数中接受 IP 地址。在以下情况下，您无需激活此功能：

- 您的用户只能使用经过身份验证的 API 操作（例如）登录 AdminInitiateAuth ，并且您使用`ContextData`参数。
- 您只想让未经身份验证的 API 操作向 Amazon Cognito 威胁防护发送设备指纹，而不是 IP 地址。

在 Amazon Cognito 控制台中按如下方式更新应用程序客户端，以添加对其他用户上下文数据的支持。

1. 登录 Amazon Cognito 控制台。
2. 在导航窗格中，选择 管理您的用户池，然后选择要编辑的用户池。
3. 选择 “应用程序客户端” 菜单。
4. 选择或创建应用程序客户端。有关更多信息，请参阅配置用户池应用程序客户端。
5. 从应用程序客户端信息容器中选择编辑。
6. 在应用程序客户端的高级身份验证设置下，选择接受其他用户上下文数据。
7. 选择保存更改。

要将您的应用程序客户端配置为接受 Amazon Cognito API 中的用户情境数据，请在CreateUserPoolClient或`EnablePropagateAdditionalUserContextData`UpdateUserPoolClient请求`true`中设置为。有关如何在 Web 或移动应用程序中使用威胁防护的信息，请参阅在应用程序中收集威胁防护的数据。当您的应用程序从服务器调用 Amazon Cognito 时，从客户端收集用户上下文数据。以下是使用 JavaScript SDK 方法的示例`getData`。

```
var EncodedData =
 AmazonCognitoAdvancedSecurityData.getData(username, userPoolId, clientId);
```

在设计应用程序以使用自适应身份验证时，我们建议您将最新的 Amazon Cognito 开发工具包集成到应用程序中。最新版本的开发工具包收集设备指纹信息，如设备 ID、模型和时区。有关 Amazon Cognito 的更多信息 SDKs，请参阅安装用户池软件开发工具包。Amazon Cognito 威胁防护仅保存您的应用程序以正确格式提交的事件并为其分配风险评分。如果 Amazon Cognito 返回错误响应，请检查您的请求是否包含有效的密钥哈希以及该`IPaddress`参数是否有效 IPv4 或 IPv6 地址。

`ContextData` 和 `UserContextData` 资源

- Amazon Amplify 适用于 Android 的 SDK：GetUserContextData
- Amazon Amplify 适用于 iOS 的 SDK：userContextData
- JavaScript: amazon-cognito-advanced-security- data.min.js

查看和导出用户事件历史记录

当您启用威胁防护时，Amazon Cognito 会为用户的每个身份验证事件生成日志。默认情况下，您可以在 Amazon Cognito 控制台的 “用户” 菜单中或AdminListUserAuthEvents通过 API 操作查看用户日志。您也可以将这些事件导出到外部系统，例如 CloudWatch 日志、Amazon S3 或 Amazon Data Firehose。导出特征使您自己的安全分析系统更容易访问有关应用程序中用户活动的安全信息。

主题

- 查看用户事件历史记录（Amazon Web Services Management Console）
- 查看用户事件历史记录（API/CLI）
- 导出用户身份验证事件

查看用户事件历史记录（Amazon Web Services Management Console）

要查看用户的登录历史记录，您可以从 Amazon Cognito 控制台的用户菜单中选择该用户。Amazon Cognito 会将用户事件历史记录保留两年。

Date (UTC)	Event	Result	Risk level	Risk decision	Challenge	IP	Device	Location	Event feedback
Jan 23, 2018 11:43:05 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 23, 2018 11:42:14 PM	Sign In	Pass	-	No Risk	Password:Success	52.94.36.11	Chrome, Windows 10	London	-
Jan 18, 2018 9:21:21 PM	Sign In	Fail	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:20:28 PM	Sign In	In Progress	High	Account Takeover	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	-
Jan 18, 2018 9:18:18 PM	Sign In	Pass	-	No Risk	Password:Success	67.132.130.174	Chrome Mobile, Android Mobile	Seattle	Invalid

5 per page < 1 2 3 >

每个登录事件都有一个事件 ID。该事件还包含对应的上下文数据，如位置、设备详细信息和风险检测结果。

您还可以将事件 ID 与 Amazon Cognito 在记录事件时发放的令牌关联起来。ID 和访问令牌在其有效负载中包含此事件 ID。Amazon Cognito 还将刷新令牌的使用与原始事件 ID 相关联。您可以通过原始事件 ID 追溯到导致颁发 Amazon Cognito 令牌的登录事件的事件 ID。您可以跟踪系统中的令牌在特定身份验证事件中的使用。有关更多信息，请参阅 了解用户池 JSON 网络令牌 (JWTs)。

查看用户事件历史记录（API/CLI）

您可以使用 Amazon Cognito API 操作AdminListUserAuthEvents或带有-events 的 Amazon Command Line Interface (Amazon CLI) 来查询用户事件历史记录。admin-list-user-auth

AdminListUserAuthEvents request

以下请求正文 `AdminListUserAuthEvents` 返回一个用户的最新活动日志。

```
{
  "UserPoolId": "us-west-2_EXAMPLE",
  "Username": "myexampleuser",
  "MaxResults": 1
}
```

admin-list-user-auth-events request

以下请求 `admin-list-user-auth-events` 返回一个用户的最新活动日志。

```
aws cognito-idp admin-list-user-auth-events --max-results 1 --username myexampleuser
 --user-pool-id us-west-2_EXAMPLE
```

Response

Amazon Cognito 对两个请求返回相同的 JSON 响应正文。以下是未发现包含风险因素的托管登录登录事件的响应示例：

```
{
    "AuthEvents": [
        {
            "EventId": "[event ID]",
            "EventType": "SignIn",
            "CreationDate": "[Timestamp]",
            "EventResponse": "Pass",
            "EventRisk": {
                "RiskDecision": "NoRisk",
                "CompromisedCredentialsDetected": false
            },
            "ChallengeResponses": [
                {
                    "ChallengeName": "Password",
                    "ChallengeResponse": "Success"
                }
            ],
            "EventContextData": {
                "IpAddress": "192.168.2.1",
                "DeviceName": "Chrome 125, Windows 10",
                "Timezone": "-07:00",
                "City": "Bellevue",
                "Country": "United States"
            }
        }
    ],
    "NextToken": "[event ID]#[Timestamp]"
}
```

导出用户身份验证事件

配置您的用户池以将用户事件从威胁防护导出到外部系统。支持的外部系统（Amazon S3、CloudWatch Logs 和 Amazon Data Firehose）可能会增加您发送或检索的数据 Amazon 账单的费用。有关更多信息，请参阅 导出威胁防护用户活动日志。

Amazon Web Services Management Console

1. 登录 Amazon Cognito 控制台。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “日志流媒体” 菜单。选择编辑。
5. 在日志记录状态下，选中激活用户活动日志导出旁边的复选框。
6. 在日志目标下，选择要处理日志 Amazon Web Services 服务 的：CloudWatch 日志组、Amazon Data Firehose 流或 S3 存储桶。
7. 您的选择将使用相应的资源类型填入资源选择器。从列表中选择日志组、流或存储桶。您也可以选择创建按钮，导航到所选服务的 Amazon Web Services Management Console 并创建新资源。
8. 选择保存更改。

API

为您的用户活动日志选择一种目标类型。

以下是将 Firehose 流设置为日志目标的示例 `SetLogDeliveryConfiguration` 请求正文。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "FirehoseConfiguration": {
            "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/
example-user-pool-activity-exported"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
```

```
}
```

以下是将 Amazon S3 存储桶设置为日志目标的示例 `SetLogDeliveryConfiguration` 请求正文。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "S3Configuration": {
            "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

以下是将 CloudWatch 日志组设置为日志目标的`SetLogDeliveryConfiguration`请求正文示例。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:DOC-
EXAMPLE-LOG-GROUP"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

提供事件反馈

事件反馈实时影响风险评估，并随着时间的推移改进风险评估算法。您可以通过 Amazon Cognito 控制台和 API 操作提供有关登录尝试有效性的反馈。

> **Note**
>
> 您的事件反馈会影响 Amazon Cognito 为具有相同特征的后续用户会话分配的风险等级。

在 Amazon Cognito 控制台中，从 “用户” 菜单中选择一个用户，然后选择 “提供事件反馈”。您可以查看事件详细信息，并选择设为有效或设为无效。

控制台在 “用户” 菜单的用户详细信息中列出了登录历史记录。您可以选择某个条目来将事件标记为有效或无效。您还可以通过用户池 API 操作和 Amazon CLI 命令 admin-update-auth-event-fe AdminUpdateAuthEventFeedbackedback 提供反馈。

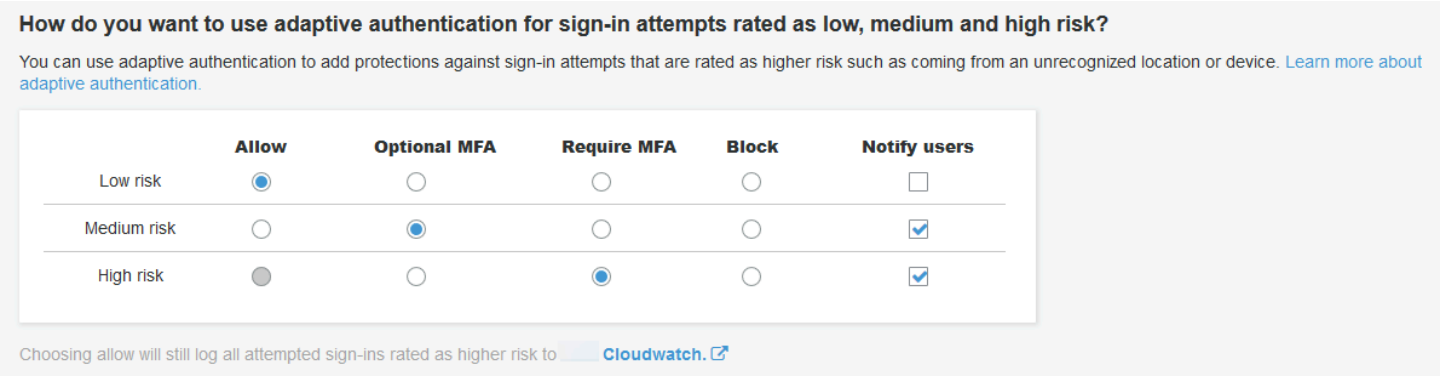
当您在 Amazon Cognito 控制台中选择设为有效或在 API 中提供 `valid` 的 `FeedbackValue` 值时，您告诉 Amazon Cognito 您信任某个用户会话（Amazon Cognito 已在其中评估了某种风险等级）。当您在 Amazon Cognito 控制台中选择设为无效或在 API 中提供 `invalid` 的 `FeedbackValue` 值时，您告诉 Amazon Cognito 您不信任某个用户会话，或者您不认为 Amazon Cognito 评估的风险等级足够高。

发送通知消息

借助威胁防护，Amazon Cognito 可以通知您的用户有风险的登录尝试。Amazon Cognito 还可以提示用户选择链接以指示登录是有效还是无效。Amazon Cognito 使用此反馈来提高用户池的风险检测准确性。

> **Note**
>
> 只有当用户的操作生成自动风险响应时，Amazon Cognito 才会向其发送通知消息：阻止登录、允许登录、将 MFA 设置为可选或要求 MFA。某些请求可能会被分配一定风险级别，但不会生成自适应身份验证自动风险响应；对于这些请求，您的用户池不会发送通知。例如，错误的密码可能会记录风险评级，但是 Amazon Cognito 的回应是登录失败，而不是应用自适应身份验证规则。

在自动风险响应部分，对低、中或高风险案例选择通知用户。



无论您的用户是否验证了电子邮件地址，Amazon Cognito 都会向他们发送电子邮件通知。

您可以自定义通知电子邮件消息，并提供这些消息的纯文本和 HTML 版本。要自定义您的电子邮件通知，请在威胁防护配置中打开自适应身份验证消息中的电子邮件模板。要了解有关电子邮件模板的更多信息，请参阅 消息模板。

在应用程序中收集威胁防护的数据

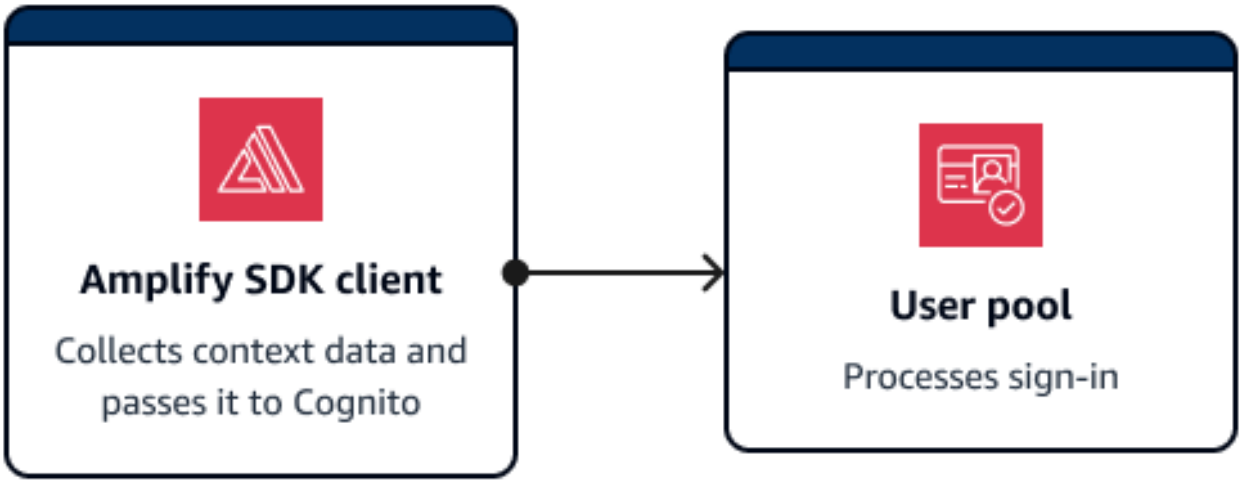
Amazon Cognito 自适应身份验证根据用户登录尝试的上下文详细信息评估账户盗用尝试的风险等级。您的应用程序必须向 API 请求添加上下文数据，这样 Amazon Cognito 威胁防护才能更准确地评估风险。上下文数据是诸如 IP 地址、浏览器代理、设备信息和请求标头之类的信息，这些信息提供有关用户如何连接到用户池的上下文信息。

向 Amazon Cognito 提交此上下文的应用程序的核心责任是向用户池发送身份验证请求中的一个 `EncodedData` 参数。要将这些数据添加到您的请求中，您可以使用软件开发工具包实现 Amazon Cognito，该软件开发工具包可以自动为您生成这些信息，也可以实现一个适用于 JavaScript iOS 或 Android 的模块来收集这些数据。必须实现直接向 Amazon Cognito 发出请求的@@ 仅限客户端的应用程序。 Amazon Amplify SDKs具有中间服务器或 API 组件的客户端-服务器应用程序必须实施单独的 SDK 模块。

在以下情况下，您的身份验证前端无需任何额外配置即可管理用户上下文数据的收集：

- 托管登录会自动收集上下文数据并将其提交给威胁防护。
- 所有 Amazon Amplify 库的身份验证方法中都内置了上下文数据集合。

使用 Amplify 在仅限客户端的应用程序中提交用户上下文数据



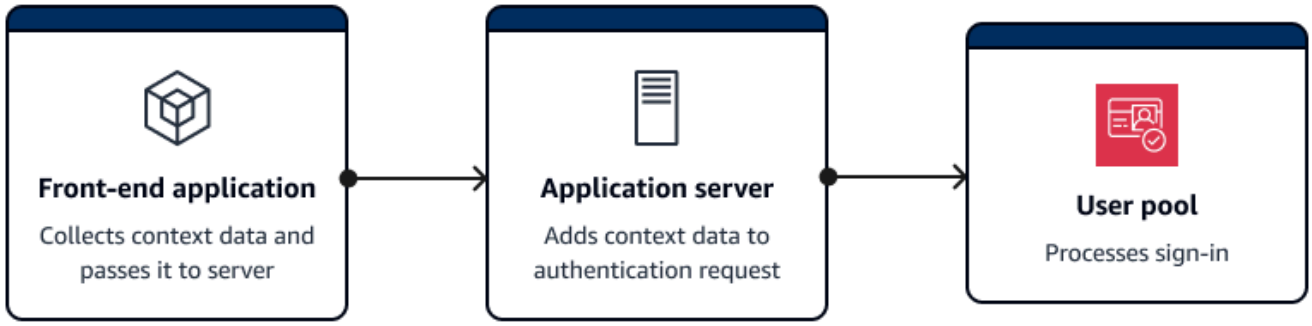
Amplify SDKs 支持直接通过 Amazon Cognito 进行身份验证的移动客户端。此类客户端直接向 Amazon Cognito 公共 API 操作发出 API 请求。默认情况下，Amplify 客户端会自动收集情境数据以进行威胁防护。

带的 Amplify 应用程序是个 JavaScript 例外。它们需要添加一个用于收集用户上下文数据的JavaScript模块。

通常，此配置中的应用程序使用未经身份验证的 API 操作，例如InitiateAuth和。RespondToAuthChallenge该UserContextData对象有助于更准确地评估这些操作的风险。Amplify SDKs 将设备和会话信息添加到的`EncodedData`参数中。`UserContextData`

在客户端-服务器应用程序中收集上下文数据

某些应用程序的前端层用于收集用户身份验证数据，应用程序的后端层用于将身份验证请求提交到 Amazon Cognito。在由微服务支持的 Web 服务器和应用程序中，这是一种常见的架构。在这些应用程序中，必须导入公共上下文数据收集库。



通常，此配置中的应用服务器使用经过身份验证的 API 操作，例如AdminInitiateAuth和AdminRespondToAuthChallenge。该ContextData对象可帮助 Amazon Cognito 更准确地评估这些操作的风险。`ContextData` 的内容是您的前端传递给服务器的编码数据，以及用户向服务器发出的 HTTP 请求中的其他详细信息。这些额外的上下文详细信息（例如 HTTP 标头和 IP 地址）为您的应用程序服务器提供了用户环境的特性。

您的应用服务器也可能使用未经身份验证的 API 操作（如InitiateAuth和）进行登录。RespondToAuthChallenge该UserContextData对象为这些操作中的威胁防护风险分析提供信息。可用的公共上下文数据收集库中的操作将安全信息添加到身份验证请求的 `EncodedData` 参数中。此外，将您的用户池配置为接受其他上下文数据，并将用户的源 IP 添加到 `UserContextData` 的 `IpAddress` 参数中。

将上下文数据添加到客户端-服务器应用程序

1. 在您的前端应用程序中，使用 iO S、Android 或 JavaScript 模块从客户端收集经过编码的上下文数据。
2. 将编码后的数据和身份验证请求的详细信息传递给应用程序服务器。
3. 在应用程序服务器中，从 HTTP 请求中提取用户的 IP 地址、相关 HTTP 标头、请求的服务器名称和请求的路径。将这些值填充到你向 Amazon Cognito 发出 API 请求的ContextData参数中。
4. 使用您的 SDK 模块收集的已编码设备数据填充您的 API 请求中的 `ContextData` 的 `EncodedData` 参数。将此上下文数据添加到身份验证请求中。

客户端-服务器应用程序的上下文数据库

JavaScript

`amazon-cognito-advanced-security-data.min.js` 模块收集了 `EncodedData`，您可以将其传递给应用程序服务器。

将该`amazon-cognito-advanced-security-data.min.js`模块添加到您的 JavaScript 配置中。`<region>`替换为以下 Amazon Web Services 区域 列表中的：`us-east-1`、`us-east-2`、`us-west-2``eu-west-1`、`eu-west-2`、或`eu-central-1`。

```
<script src="https://amazon-cognito-assets.<region>.amazoncognito.com/amazon-cognito-advanced-security-data.min.js"></script>
```

要生成可在`EncodedData`参数中使用的`encodedContextData`对象，请将以下内容添加到您的 JavaScript 应用程序源中：

```
var encodedContextData = AmazonCognitoAdvancedSecurityData.getData(_username,
 _userpoolId, _userPoolClientId);
```

iOS/Swift

要生成上下文数据，iOS 应用程序可以集成适用于 iO S 的移动 SDK 模块 AWSCognitoIdentityProviderASF。

要收集经过编码的上下文数据以进行威胁防护，请将以下代码段添加到您的应用程序中：

```
import AWSCognitoIdentityProviderASF

let deviceId = getDeviceId()
let encodedContextData = AWSCognitoIdentityProviderASF.userContextData(
                            userPoolId,
                            username: username,
                            deviceId: deviceId,
                            userPoolClientId: userPoolClientId)

/**
 * Reuse DeviceId from keychain or generate one for the first time.
 */
func getDeviceId() -> String {
```

```
    let deviceIdKey = getKeyChainKey(namespace: userPoolId, key:
 "AWSCognitoAuthAsfDeviceId")

   if let existingDeviceId = self.keychain.string(forKey: deviceIdKey) {
        return existingDeviceId
    }

    let newDeviceId = UUID().uuidString
    self.keychain.setString(newDeviceId, forKey: deviceIdKey)
    return newDeviceId
}

/**
 * Get a namespaced keychain key given a namespace and key
 */
func getKeyChainKey(namespace: String, key: String) -> String {
    return "\(namespace).\(key)"
}
```

Android

要生成上下文数据，安卓应用可以集成适用于安卓的移动 SDK 模块 aws-android-sdk-cognitoidentityprovider-asf。

要收集经过编码的上下文数据以进行威胁防护，请将以下代码段添加到您的应用程序中：

```
UserContextDataProvider provider = UserContextDataProvider.getInstance();
// context here is android application context.
String encodedContextData = provider.getEncodedContextData(context, username,
 userPoolId, userPoolClientId);
```

将 Amazon WAF Web ACL 与用户池关联

Amazon WAF 是 Web 应用程序防火墙。借助 Amazon WAF 网络访问控制列表 (Web ACL)，您可以保护您的用户池免受对经典托管用户界面和 Amazon Cognito API 服务终端节点的不必要请求的影响。Web ACL 使您可以对用户池响应的所有 HTTPS Web 请求进行精细控制。有关 Amazon WAF Web 的更多信息 ACLs，请参阅《Amazon WAF 开发人员指南》中的管理和使用 Web 访问控制列表 (Web ACL)。

当您的 Amazon WAF Web ACL 与用户池关联时，Amazon Cognito 会将您的用户请求中选定的非机密标头和内容转发到。 Amazon WAF Amazon WAF 检查请求的内容，将其与您在网页 ACL 中指定的规则进行比较，然后向 Amazon Cognito 返回响应。

关于 Amazon WAF 网络 ACLs 和亚马逊 Cognito 的注意事项

- 目前，Web ACL 规则仅适用于向使用托管 UI（经典）品牌版本的用户池域发出的请求。当您设置`ManagedLoginVersion`为`2`托管登录或您的品牌版本设置为托管登录时，Amazon Cognito 不会在您的托管登录页面上强制执行规则。

  要将您的品牌版本更改为与 Amazon WAF 网络兼容 ACLs，请执行以下任一操作。此更改会影响您的登录页面的外观和功能。

  - 在CreateUserPoolDomain或 UpdateUserPoolDomainAPI 请求中，设置`ManagedLoginVersion`为`1`。
  - 在 Amazon Cognito 控制台用户池的域名菜单中，编辑您的前缀或经典域，并将托管登录版本设置为托管用户界面（经典）。

  有关品牌版本的更多信息，请参阅用户池托管登录。
- 您无法将 Web ACL 规则配置为匹配用户池请求中的个人身份信息 (PII)，例如用户名、密码、电话号码或电子邮件地址。此数据将不可用 Amazon WAF。相反，请将 Web ACL 规则配置为匹配标头、路径和正文中的会话数据，例如 IP 地址、浏览器代理和请求的 API 操作。
- 被阻止的请求 Amazon WAF 不计入任何请求类型的请求速率配额。该 Amazon WAF 处理程序在 API 级别的限制处理程序之前被调用。
- 当您创建 Web ACL 时，Web ACL 要经过一小段时间才能完全传播并可供 Amazon Cognito 使用。传播时间可以从几秒钟到几分钟不等。 Amazon WAF `WAFUnavailableEntityException`当您尝试在 Web ACL 完全传播之前将其关联时，将返回。
- 您可以将一个 Web ACL 与一个用户池关联。
- 您的请求可能会导致负载超出 Amazon WAF 可以检查的负载限制。请参阅Amazon WAF 开发者指南中的超大请求组件处理，了解如何配置如何 Amazon WAF 处理来自 Amazon Cognito 的超大请求。
- 您无法将使用防 Amazon WAF 欺诈控制账户盗用 (ATP) 的网络 ACL 与 Amazon Cognito 用户池相关联。在添加 `AWS-AWSManagedRulesATPRuleSet` 托管规则组时实施 ATP 功能。在将您的 Web ACL 与用户池关联之前，请确保该 Web ACL 不使用此托管规则组。
- 当你有一个 Amazon WAF Web ACL 与用户池相关联，并且你的 Web ACL 中的规则显示了验证码时，这可能会导致经典托管 UI TOTP 注册中出现无法恢复的错误。要创建具有验证码操作且不影响经典托管用户界面 TOTP 的规则，请参阅。为托管登录 TOTP MFA 配置您的 Amazon WAF 网页 ACL

Amazon WAF 检查对以下端点的请求。

经典托管用户界面

对 用户池端点和托管登录参考 中所有端点的请求。

公有 API 操作

您的应用程序向 Amazon Cognito API 发出的不使用 Amazon 凭证进行授权的请求。这包括InitiateAuthRespondToAuthChallenge、和 API 操作GetUser。范围内的 API 操作 Amazon WAF 不需要使用 Amazon 凭据进行身份验证。它们未经身份验证，或者是使用会话字符串或访问令牌授权的。有关更多信息，请参阅 按授权模型分组的 API 操作列表。

您可以使用以下规则操作在 Web ACL 中配置规则：计数、允许、阻止，或者提供一个验证码以响应符合规则的请求。有关更多信息，请参阅 Amazon WAF 开发人员指南 中的 Amazon WAF 规则。根据规则操作，您可以自定义 Amazon Cognito 返回给用户的响应。

> **Important**
>
> 自定义错误响应的选项取决于您发出 API 请求的方式。
>
> - 您可以自定义经典托管界面请求的错误代码和响应正文。您只能在经典的托管用户界面中提供验证码供用户解析。
> - 对于您使用 Amazon Cognito 用户池 API 提出的请求，可以自定义收到阻止响应的请求的响应正文。您也可以指定 400–499 范围内的自定义错误代码。
> - Amazon Command Line Interface (Amazon CLI) 和向生成区块或验证码响应的请求 Amazon SDKs 返回`ForbiddenException`错误。

将 Web ACL 与您的用户池相关联

要在您的用户池中使用网络 ACL，您的 Amazon Identity and Access Management (IAM) 委托人必须具有以下 Amazon Cognito 和 Amazon WAF 权限。有关 Amazon WAF 权限的信息，请参阅《Amazon WAF 开发者指南》中的 Amazon WAF API 权限。

```
{
 "Version": "2012-10-17",
 "Statement": [
  {
   "Sid": "AllowWebACLUserPool",
   "Effect": "Allow",
   "Action": [
```

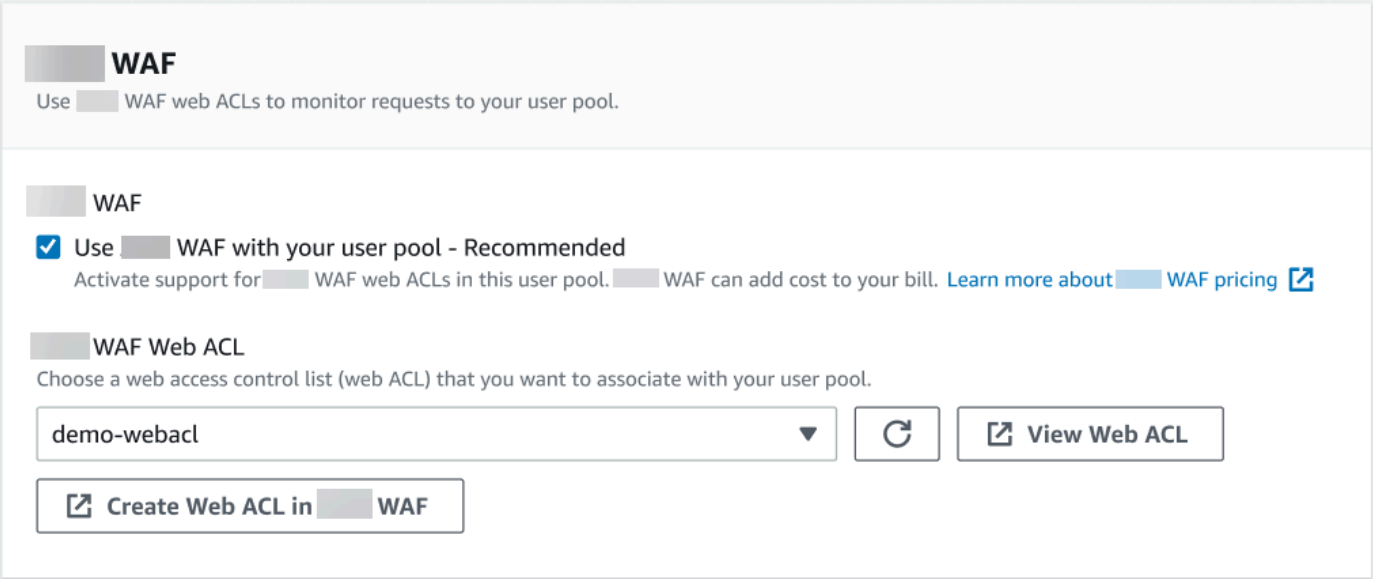
```
      "cognito-idp:ListResourcesForWebACL",
      "cognito-idp:GetWebACLForResource",
      "cognito-idp:AssociateWebACL"
     ],
     "Resource": [
      "arn:aws:cognito-idp:*:123456789012:userpool/*"
     ]
    },
    {
     "Sid": "AllowWebACLUserPoolWAFv2",
     "Effect": "Allow",
     "Action": [
      "wafv2:ListResourcesForWebACL",
      "wafv2:AssociateWebACL",
      "wafv2:DisassociateWebACL",
      "wafv2:GetWebACLForResource"
     ],
     "Resource": "arn:aws:wafv2:*:123456789012:*/webacl/*/*"
    },
    {
     "Sid": "DisassociateWebACL1",
     "Effect": "Allow",
     "Action": "wafv2:DisassociateWebACL",
     "Resource": "*"
    },
    {
     "Sid": "DisassociateWebACL2",
     "Effect": "Allow",
     "Action": [
      "cognito-idp:DisassociateWebACL"
     ],
     "Resource": [
      "arn:aws:cognito-idp:*:123456789012:userpool/*"
     ]
    }
   ]
}
```

尽管您必须授予 IAM 权限，但列出的操作仅限权限，与任何 API 操作都不对应。

激活 Amazon WAF 您的用户池并关联 Web ACL

1. 登录 Amazon Cognito 控制台。

2. 在导航窗格中，选择 用户池，然后选择要编辑的用户池。
3. 在 “安全” 部分中选择该Amazon WAF选项卡。
4. 选择编辑。
5. 选择 “ Amazon WAF 与您的用户池一起使用”。



6. 选择您已经创建的Amazon WAF Web ACL，或者选择在中创建 Web ACL Amazon WAF，在中的新 Amazon WAF 会话中创建一个 Web ACL Amazon Web Services Management Console。
7. 选择保存更改。

要以编程方式将 Web ACL 与 Amazon Command Line Interface 或 SDK 中的用户池关联，请使用 Amazon WAF API AssociateWeb中的 ACL。Amazon Cognito 没有可关联 Web ACL 的单独 API 操作。

测试和记录 Amazon WAF 网页 ACLs

当您在 Web ACL 中将规则操作设置为 C ou nt 时， Amazon WAF 会将该请求添加到与该规则匹配的请求计数中。要使用您的用户池测试 Web ACL，请将规则操作设置为计数，并考虑与每条规则匹配的请求数量。例如，如果您要设置为阻止操作的规则与您确定为正常用户流量的大量请求相匹配，则您可能需要重新配置规则。有关更多信息，请参阅《Amazon WAF 开发人员指南》中的测试和调整您的 Amazon WAF 保护。

您还可以配置 Amazon WAF 为将请求标头记录到亚马逊 CloudWatch 日志组、亚马逊简单存储服务 (Amazon S3) 存储桶或亚马逊数据 Firehose。您可以通过 `x-amzn-cognito-client-id` 和 `x-amzn-cognito-operation-name` 识别您使用用户池 API 发出的 Amazon Cognito 请求。托管的 UI

请求仅包含 `x-amzn-cognito-client-id` 标头。有关更多信息，请参阅 Amazon WAF 开发人员指南 中的记录 Web ACL 流量。

Amazon WAF web ACLs 在所有用户池功能计划中都可用。的安全功能 Amazon WAF 补充了 Amazon Cognito 威胁防护。您可以在用户池中激活这两项功能。 Amazon WAF 对用户池请求的检查单独收费。有关更多信息，请参阅Amazon WAF 定价。

日志 Amazon WAF 请求数据需要按您定位日志的服务收取额外费用。有关更多信息，请参阅 Amazon WAF 开发人员指南 中的记录 Web ACL 流量信息的定价。

用户池区分大小写

默认情况下，您在中创建的 Amazon Cognito 用户池不区分大小写。 Amazon Web Services Management Console 当用户池不区分大小写时，user@example.com 和 User@example.com 指的是同一个用户。当用户池中的用户名不区分大小写时，`preferred_username` 和 `email` 属性也不区分大小写。

要考虑用户池区分大小写设置，请根据备用用户属性识别应用程序代码中的用户。由于用户名、首选用户名或电子邮件地址属性的大小写在不同的用户配置文件中可能有所不同，因此请改为参阅 `sub` 属性。您还可以在用户池中创建不可变的自定义属性，并为每个新用户配置文件中的属性分配自己的唯一标识符值。首次创建用户时，可以将值写入您创建的不可改变的自定义属性。

> **Note**
>
> 无论您的用户池的区分大小写设置如何，Amazon Cognito 都要求来自 SAML 或 OIDC 身份提供商（IdP）的联合用户传递唯一且区分大小写的 `NameId` 或 `sub` 声明。有关唯一标识符区分大小写和 SAML 的更多信息 IdPs，请参阅实现 SP 启动的 SAML 登录。

创建区分大小写的用户池

如果您使用 Amazon Command Line Interface (Amazon CLI) 和 API 操作（例如）创建资源 CreateUserPool，则必须将布尔`CaseSensitive`参数设置为`false`。此设置将创建不区分大小写的用户池。如果您不指定值，`CaseSensitive` 默认为 `true`。您在 Amazon Cognito 控制台中创建的用户池区分大小写。要生成区分大小写的用户池，必须使用`CreateUserPool`操作。2020 年 2 月 12 日之前，无论平台如何，用户池都默认区分大小写。

在 Amazon Web Services Management Console 和`UsernameConfiguration`属性的登录菜单中 DescribeUserPool，您可以查看账户中每个用户池的区分大小写设置。

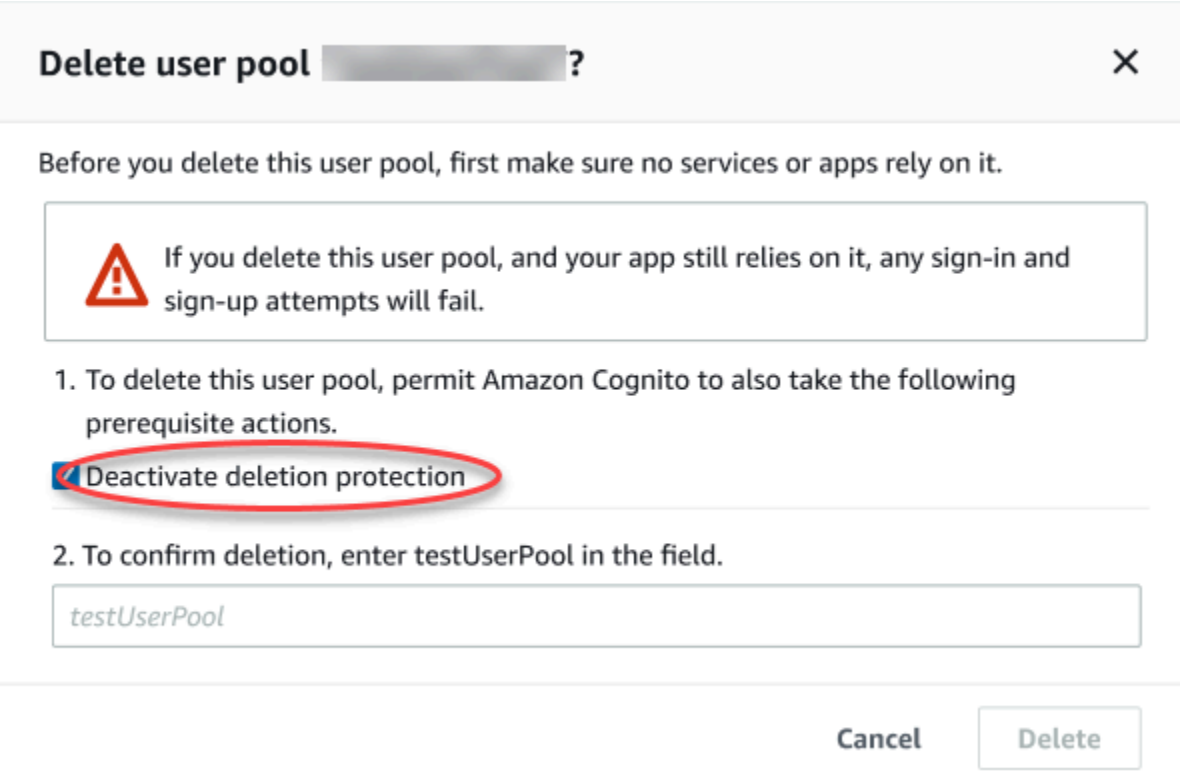
迁移到新的用户池

由于用户配置文件之间存在潜在冲突，您无法将 Amazon Cognito 用户池从区分大小写更改为不区分大小写。相反，请将用户迁移到新的用户池。您必须构建迁移代码才能解决与大小写相关的冲突。此代码必须返回唯一的新用户，或者在检测到冲突时拒绝登录尝试。在新的不区分大小写的用户池中，分配一个 迁移用户 Lambda 触发器。该 Amazon Lambda 函数可以在新的不区分大小写的用户池中创建用户。当用户使用不区分大小写的用户池登录失败时，Lambda 函数会从区分大小写的用户池中查找并复制该用户。您还可以针对事件激活迁移用户 Lambda 触发器。ForgotPasswordAmazon Cognito 会将用户信息和事件元数据从登录或密码恢复操作传递到您的 Lambda 函数。当函数在不区分大小写的用户池中创建新用户时，您可以使用事件数据来管理用户名和电子邮件地址之间的冲突。这些冲突发生在用户名和电子邮件地址之间，这些冲突在区分大小写的用户池中是唯一的，但在不区分大小写的用户池中是相同的。

有关如何在 Amazon Cognito 用户池之间使用迁移用户 Lambda 触发器的更多信息，请参阅博客中的将用户迁移到 Amazon Cognito 用户池。 Amazon

用户池删除保护

要使您的管理员不会意外删除用户池，请激活删除保护。启用删除保护后，必须先确认要删除用户池，然后才能将其删除。在中删除用户池时 Amazon Web Services Management Console，可以同时停用删除保护。当您接受停用删除保护的提示并确认要删除时（如下图所示），Amazon Cognito 会删除您的用户池。



如果您想通过 Amazon Cognito API 请求删除用户池，则必须先在请求`Inactive`中`DeletionProtection`更改为UpdateUserPool。如果您不停用删除保护，Amazon Cognito 会返回 `InvalidParameterException` 错误。停用删除保护后，可以在DeleteUserPool请求中删除用户池。

当您在 Amazon Web Services Management Console中创建新的用户群体时，Amazon Cognito 默认情况下会激活 Deletion protection（删除保护）。当您使用 `CreateUserPool` API 创建用户群体时，默认情况下未激活删除保护。要在使用 Amazon CLI 或 Amazon SDK 创建的用户池中使用此功能，请将`DeletionProtection`参数设置为`True`。

您可以在 Amazon Cognito 控制台的 “设置” 菜单的删除保护容器中激活或停用删除保护状态。

配置删除保护

1. 转到 Amazon Cognito 控制台。系统可能会提示您输入 Amazon 凭证。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “设置” 菜单并导航到 “删除保护” 选项卡。选择 “激活” 或 “停用”。
5. 在下一个对话框中确认您的选择。

管理用户存在错误响应

Amazon Cognito 支持自定义由用户群体返回的错误响应。自定义错误响应可用于用户创建和身份验证、密码恢复和确认操作。

使用用户池应用程序客户端的 PreventUserExistenceErrors 设置，以启用或禁用用户存在相关错误。当您使用 Amazon Cognito 用户池 API 创建新的应用程序时，默认情况下 PreventUserExistenceErrors 为 LEGACY 或禁用。在 Amazon Cognito 控制台中，默认选定防止用户已存在错误选项（PreventUserExistenceErrors 设置为 ENABLED）。要更新 PreventUserExistenceErrors 配置，请执行以下操作之一：

- 在 UpdateUserPoolClient API 请求中将 PreventUserExistenceErrors 的值更改为 ENABLED 和 LEGACY。
- 在 Amazon Cognito 控制台中编辑应用程序客户端，并将防止用户已存在错误的状态更改为选定（ENABLED）和已取消选择（LEGACY）。

当此属性的值为 LEGACY 时，当用户尝试使用您的用户池中不存在的用户名登录时，应用程序客户端会返回 UserNotFoundException 错误响应。

当此属性的值为 ENABLED 时，应用程序客户端不会通过 UserNotFoundException 错误来透露您的用户池中不存在某个用户账户。当您提交用户名不存在的请求时，的PreventUserExistenceErrors配置会产生以下影响：ENABLED

- Amazon Cognito 会使用非特定信息响应 API 请求，否则其响应可能会泄露存在有效的用户。
- Amazon Cognito 会对忘记密码的请求以及除基于选择的身份验证 () 之外的身份验证流程的身份验证请求返回一般的身份验证失败响应，例如或。USER_AUTH USER_SRP_AUTH CUSTOM_AUTH错误响应告知您用户名或密码不正确。
- Amazon Cognito 通过从用户池允许的质询类型中随机选择来回应基于选择的身份验证请求。您的用户池可能会返回密钥、一次性密码或密码质询。
- Amazon Cognito 账户确认和密码恢复会 APIs 返回一个响应，表示代码已发送到模拟传送媒体，而不是部分显示用户的联系信息。

以下信息详细说明了 PreventUserExistenceErrors 设置为 ENABLED 时用户池操作的行为。

身份验证和用户创建操作

可以在用户名密码和安全远程密码 (SRP) 身份验证中配置错误响应。您还可以对使用自定义身份验证返回的错误进行自定义。基于选择的身份验证不受您的配置的影响。PreventUserExistenceErrors

身份验证流程中的用户存在性披露细节

基于选择的身份验证

在USER_AUTH基于选择的身份验证流程中，Amazon Cognito 会根据可用的主要身份验证因素返回质询，具体取决于您的用户池配置和用户属性。此身份验证流程可以返回密码、安全远程密码 (SRP)、 WebAuthn (passkey)、SMS 一次性密码 (OTP) 或电子邮件 OTP 质询。PreventUserExistenceErrors激活后，Amazon Cognito 会向不存在的用户发出挑战，要求他们完成一种或多种可用形式的身份验证。PreventUserExistenceErrors处于非活动状态时，Amazon Cognito 会返回异常。UserNotFound

用户名和密码身份验证

身份验证流ADMIN_USER_PASSWORD_AUTHUSER_PASSWORD_AUTH、的PASSWORD流程在处于活动状态Incorrect username or password时USER_AUTH返回NotAuthorizedException带有消息PreventUserExistenceErrors的a。当PreventUserExistenceErrors处于非活动状态时，这些流将返回UserNotFoundException。

基于安全远程密码（SRP）的身份验证

作为最佳实践，仅在没有电子邮件地址、电话号码USER_SRP_AUTH或首选用户名别名属性的用户池USER_AUTH中实现 PreventUserExistenceErrors with 或 the flo PASSWORD_SRP w。在 SRP 身份验证流程中，具有别名属性的用户可能不会受到禁止用户存在的限制。用户名密码身份验证流程 — ADMIN_USER_PASSWORD_AUTH USER_PASSWORD_AUTH、和USER_AUTHPASSWORD质询 — 完全禁止用户通过别名属性存在。

当有人尝试使用您的应用程序客户端不知道的用户名登录 SRP 时，Amazon Cognito 会在第一步中返回模拟响应，如 RFC 5054 中所述。Amazon Cognito 始终针对相同的用户名和用户池组合返回相同的盐值以及 UUID 格式的内部用户 ID。当您发送带有密码证明的 RespondToAuthChallenge API 时，Amazon Cognito 在用户名或密码不正确时返回一个通用 NotAuthorizedException 错误。有关实现 SRP 身份验证的更多信息，请参阅使用永久密码和安全负载登录。

> **Note**
>
> 如果您使用基于验证的别名属性，并且不可改变的用户名格式不是 UUID，则可以模拟使用用户名和密码身份验证的通用响应。

自定义身份验证质询 Lambda 触发器

当用户尝试使用身份验证流程登录但未找到其用户名时，Amazon Cognito 会调用 Lambda 触发的CUSTOM_AUTH自定义身份验证质询。true对于任何不存在的用户，输入事件都UserNotFound包含一个名为的布尔参数，其值为。此参数出现在您的用户池发送给构成自定义身份验证架构的创建、定义和验证身份验证质询 Lambda 函数的请求事件中。当您在 Lambda 函数的逻辑中检查此指标时，您可以为不存在的用户模拟自定义身份验证挑战。

身份验证前 Lambda 触发器

当用户尝试登录但找不到用户名时，Amazon Cognito 会调用身份验证前触发器。true对于任何不存在的用户，输入事件都包含一个值为的UserNotFound参数。

以下列表描述了对PreventUserExistenceErrors用户帐户创建的影响。

用户创建流程中的用户存在披露细节

SignUp

当已使用用户名时，SignUp 操作始终返回 UsernameExistsException。如果在您的应用程序中注册用户时，您不希望 Amazon Cognito 为电子邮件地址和电话号码返回 UsernameExistsException 错误，请使用基于验证的别名属性。有关别名的更多信息，请参阅自定义登录属性。

有关 Amazon Cognito 如何阻止使用 SignUp API 请求来发现用户群体中用户的示例，请参阅 在注册时防止出现电子邮件地址和电话号码的 UsernameExistsException 错误。

导入的用户

如果 PreventUserExistenceErrors 已启用，则在对导入的用户进行身份验证期间，将返回通用 NotAuthorizedException 错误，指示用户名或密码不正确，而不是返回 PasswordResetRequiredException。请参阅要求导入的用户重置密码了解更多信息。

迁移用户 Lambda 触发器

当 Lambda 触发器在原始事件上下文中设置了空响应时，Amazon Cognito 将为不存在的用户返回模拟响应。有关更多信息，请参阅 利用用户迁移 Lambda 触发器导入用户。

在注册时防止出现电子邮件地址和电话号码的 **UsernameExistsException** 错误

以下示例演示了在用户群体中配置别名属性时，如何在对 SignUp API 请求的响应中，防止重复的电子邮件地址和电话号码生成 UsernameExistsException 错误。您必须在创建用户群体时使用电子邮件地址或电话号码作为别名属性。有关更多信息，请参阅用户群体属性的自定义登录属性 部分。

1. Jie 注册了一个新的用户名，还提供了电子邮件地址 jie@example.com。Amazon Cognito 将向其电子邮件地址发送一个代码。

   Amazon CLI 命令示例

   ```
   aws cognito-idp sign-up --client-id 1234567890abcdef0 --username jie --password
    PASSWORD --user-attributes Name="email",Value="jie@example.com"
   ```

   响应示例

   ```
   {
       "UserConfirmed": false,
       "UserSub": "<subId>",
       "CodeDeliveryDetails": {
           "AttributeName": "email",
           "Destination": "j****@e****",
           "DeliveryMedium": "EMAIL"
       }
   }
   ```

2. Jie 提供了发送过来的代码，确认其拥有该电子邮件地址。这样就完成了用户注册。

   Amazon CLI 命令示例

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=jie --
   confirmation-code xxxxxx
   ```

3. Shirley 注册了一个新的用户账户并提供了电子邮件地址 jie@example.com。Amazon Cognito 不会返回 UsernameExistsException 错误，而是向 Jie 的电子邮件地址发送确认代码。

Amazon CLI 命令示例

```
aws cognito-idp sign-up --client-id 1234567890abcdef0 --username shirley --password
 PASSWORD --user-attributes Name="email",Value="jie@example.com"
```

响应示例

```
{
    "UserConfirmed": false,
    "UserSub": "<new subId>",
    "CodeDeliveryDetails": {
        "AttributeName": "email",
        "Destination": "j****@e****",
        "DeliveryMedium": "EMAIL"
    }
}
```

4. 在另一种情况下，Shirley 拥有对 `jie@example.com` 的所有权。Shirley 收到了 Amazon Cognito 发送到 Jie 电子邮件地址的代码，并尝试确认该账户。

   Amazon CLI 命令示例

   ```
   aws cognito-idp confirm-sign-up --client-id 1234567890abcdef0 --username=shirley --
   confirmation-code xxxxxx
   ```

   响应示例

   ```
   An error occurred (AliasExistsException) when calling the ConfirmSignUp operation: An
    account with the email already exists.
   ```

尽管已将 `jie@example.com` 分配给现有用户，Amazon Cognito 不会对 Shirley 的 `aws cognito-idp sign-up` 请求返回错误。在 Amazon Cognito 返回错误响应之前，Shirley 必须证明对该电子邮件地址的所有权。在具有别名属性的用户群体中，此行为会阻止使用公共 `SignUp` API 来检查是否存在具有给定电子邮件地址或电话号码的用户。

此行为与 Amazon Cognito 向使用现有用户名的 `SignUp` 请求返回的响应不同，如以下示例所示。尽管 Shirley 从此回复中得知已经存在具有用户名 `jie` 的用户，但他们并不知道与该用户关联的任何电子邮件地址或电话号码。

示例 CLI 命令

```
aws cognito-idp sign-up --client-id 1example23456789 --username jie --password PASSWORD
      --user-attributes Name="email",Value="shirley@example.com"
```

响应示例

```
An error occurred (UsernameExistsException) when calling the SignUp operation: User
 already exists
```

密码重置操作

当您防止出现用户存在错误时，Amazon Cognito 会对用户密码重置操作返回以下响应。

ForgotPassword

当找不到用户、已停用用户或没有经过验证的交付机制来恢复其密码时。Amazon Cognito 以不同的方式处理现有和不存在的用户。

- 现有用户-对于没有经过验证的用于账户恢复的电子邮件/电话的现有用户，Amazon Cognito 会返回。`InvalidParameterException`
- 不存在的用户-对于没有经过验证的电子邮件/电话进行账户恢复的不存在的用户，Amazon Cognito 会同时返回带有模拟传送介质的和。`InvalidParameterException` `CodeDeliveryDetails`模拟的传递媒介由用户池的输入用户名格式和验证设置决定。

ConfirmForgotPassword

Amazon Cognito 为不存在或已禁用的用户返回 `CodeMismatchException` 错误。如果在使用 `ForgotPassword` 时不请求代码，Amazon Cognito 将返回 `ExpiredCodeException` 错误。

确认操作

当您防止出现用户存在错误时，Amazon Cognito 会对用户确认和验证操作返回以下响应。

ResendConfirmationCode

Amazon Cognito 为已禁用或不存在的用户返回 `CodeDeliveryDetails`。Amazon Cognito 会向现有用户的电子邮件或电话发送确认码。

ConfirmSignUp

如果代码已过期，则将返回 `ExpiredCodeException`。当用户未被授权时，Amazon Cognito 返回 `NotAuthorizedException`。如果代码与服务器期望的代码不匹配，则 Amazon Cognito 返回 `CodeMismatchException`。

用户池端点和托管登录参考

Amazon Cognito 有两种用户池身份验证模式：使用用户池 API 和使用 OAuth 2.0 授权服务器。当你想在应用程序后端使用 Amazon 软件开发工具包检索 OpenID Connect (OIDC) 令牌时，请使用 API。当您要将用户池实施为 OIDC 提供者时，请使用授权服务器。授权服务器添加了诸如联合登录、具有访问令牌范围的 API 和 M2M 授权以及托管登录等功能。您可以单独使用 API 模型和 OIDC 模型，也可以将它们一起使用，并可以在用户池级别或应用程序客户端级别进行配置。本节提供了实施 OIDC 模型的参考。有关这两种身份验证模型的更多信息，请参阅了解 API、OIDC 和托管登录页面身份验证。

当您向用户池分配域时，Amazon Cognito 会激活此处列出的公开网页。您的域用作所有应用程序客户端的中央接入点。它们包括托管登录（您的用户可以在其中注册并登录（登录端点），以及注销（注销端点）。有关这些资源的标签的更多信息，请参阅 用户池托管登录。

这些页面还包括公共网络资源，允许您的用户池与第三方 SAML、OpenID Connect (OIDC OAuth) 和 2.0 身份提供商 () 进行通信。IdPs要使用联合身份提供商登录用户，您的用户必须向交互式托管登录登录端点或 OID 对端点授权 C 发起请求。Authorize 端点会将您的用户重定向到您的托管登录页面或 IdP 登录页面。

您的应用程序还可以使用 Amazon Cognito 用户池 API 登录本地用户。本地用户仅存在于您的用户池目录中，无需通过外部 IdP 进行联合身份验证。

除了托管登录外，Amazon Cognito 还集成了 SDKs 适用于安卓 JavaScript、iOS 等设备的管理登录。SDKs 提供了使用亚马逊 Cognito API 服务终端节点执行用户池 API 操作的工具。有关服务端点的更多信息，请参阅 Amazon Cognito 身份端点和限额。

> **Warning**
>
> 不要锁定 Amazon Cognito 域的终端实体或中间传输层安全 (TLS) 证书。 Amazon 管理所有用户池终端节点和前缀域的所有证书。信任链中支持 Amazon Cognito 证书的证书颁发机构 (CAs) 会动态轮换和续订。当您将应用程序固定到中间证书或叶证书时，您的应用程序在 Amazon 轮换证书时可能会失败，恕不另行通知。

而应将应用程序固定到所有可用的 Amazon 根证书。有关更多信息，请参阅《Amazon Certificate Manager 用户指南》的证书固定中的最佳做法和建议。

主题

- 用户交互式托管登录和经典托管用户界面端点
- 身份提供者和依赖方端点
- OAuth 2.0 补助金
- 在授权代码授予中使用 PKCE
- 托管登录和联盟错误响应

用户交互式托管登录和经典托管用户界面端点

当您将域添加到用户池时，Amazon Cognito 会激活本节中的托管登录终端节点。这些端点是一些网页，您的用户可以在这些网页中完成用户池的核心身份验证操作。它们包括用于密码管理、多重身份验证（MFA）和属性验证的页面。

构成托管登录的网页是一个前端 Web 应用程序，用于与客户进行交互式用户会话。您的应用程序必须在用户的浏览器中调用托管登录。Amazon Cognito 不支持以编程方式访问本章中的网页。可以在应用程序代码中直接查询 身份提供者和依赖方端点 中那些返回 JSON 响应的联合端点。对端点授权重定向到托管登录或 IdP 登录页面，并且还必须在用户的浏览器中打开。

所有用户池端点都接受来自 IPv4 和 IPv6 源 IP 地址的流量。

本指南中的主题详细描述了常用的托管登录和经典托管用户界面端点。托管登录和托管用户界面之间的区别是显而易见的，但不起作用。除此之外`/passkeys/add`，所有路径均在两个版本的托管登录品牌之间共享。

Amazon Cognito 会在您为用户池分配域时提供以下网页。

托管登录端点

端点 URL	描述	如何访问此端点
https://*Your user pool domain*/登录	登录用户池本地用户和联合用户。	从端点重定向，如对端点授权、`/logout` 和 `/confirmforgotPassword` 。请参阅登录端点。

端点 URL	描述	如何访问此端点
https://*Your user pool domain*/注销	注销用户池用户。	直接链接。请参阅注销端点。
https:// *Your user pool domain* /confirmus	确认已选择电子邮件链接来验证其用户账户的用户。	电子邮件中用户选择的链接。
https:// *Your user pool domain* /signu	注册一个新用户。当您的用户选择注册时，`/login` 页面会将他们定向到 `/signup`。	与 `/oauth2/authorize` 具有相同参数的直接链接。
https://*Your user pool domain*/确认	在用户池向已注册的用户发送确认码后，提示您的用户输入验证码。	仅从 `/signup` 重定向。
https:// *Your user pool domain* /忘记密码	提示您的用户输入其用户名并发送密码重置代码。在您的用户选择忘记密码？时，`/login` 页面会将他们定向到 `/forgotPassword` 。	1. 从 `/login` 中的忘记密码链接。 2. 与 `/oauth2/authorize` 具有相同参数的直接链接。
https://*Your user pool domain*/确认忘记密码	提示您的用户输入其密码重置代码和新的密码。在您的用户选择重置密码时，`/forgotPassword` 页面会将他们定向到 `/confirmforgotPassword` 。	仅从 `/forgotPassword` 重定向。
https:/ *Your user pool domain* /resendcode	向已在用户池中注册的用户发送新的确认码。	仅从 `/confirm` 中的发送新代码链接重定向。
https: ///passkeys *Your user pool domain* /add	注册新的密钥。仅在托管登录中可用。	• 在支持密钥身份验证的应用程序客户端中确认后的注册流程中。 • 与 `/oauth2/authorize` 具有相同参数的直接链接。

主题

- 托管登录终端节点：/login
- 托管登录注销端点：/logout

托管登录终端节点：/login

登录端点是身份验证服务器和来自的重定向目的地对端点授权。当您不指定身份提供商时，它是托管登录的入口点。当您生成到登录端点的重定向时，它加载登录页面，并向用户显示为客户端配置的身份验证选项。

> **Note**
>
> 登录端点是托管登录的组成部分。在您的应用程序中，调用重定向到登录端点的联盟和托管登录页面。用户直接访问登录端点并不是最佳实践。

GET /login

`/login` 端点仅支持对用户的初始请求执行 `HTTPS GET`。您的应用程序会在 Chrome 或 Firefox 等浏览器中调用该页面。当您从 对端点授权 重定向到 `/login` 时，它会传递您在初始请求中提供的所有参数。登录端点支持授权端点的所有请求参数。您也可以直接访问登录端点。作为最佳实践，应在 `/oauth2/authorize` 上发起所有用户会话。

示例 - 提示用户登录

此示例显示登录屏幕。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/login?
                response_type=code&
                client_id=ad398u21ijw3s9w3939&
                redirect_uri=https://YOUR_APP/redirect_uri&
                state=STATE&
                scope=openid+profile+aws.cognito.signin.user.admin
```

示例：响应

身份验证服务器重定向到您的应用程序并提供授权代码和状态。服务器必须在查询字符串参数中返回代码和状态，而不是在片段中。

```
HTTP/1.1 302 Found
                     Location: https://YOUR_APP/redirect_uri?
code=AUTHORIZATION_CODE&state=STATE
```

用户发起的登录请求

在用户加载 /login 端点后，他们可以输入用户名和密码并选择登录。这样做时，将生成一个与 GET 请求具有相同标头请求参数的 HTTPS POST 请求，以及包含用户名、密码和设备指纹的请求正文。

托管登录注销端点：/logout

/logout 端点是重定向端点。该端点使用户注销并重定向到您的应用程序客户端的授权注销 URL，或重定向到 /login 端点。/logout终端节点的 GET 请求中的可用参数是针对 Amazon Cognito 托管登录用例量身定制的。

注销端点是一个前端 Web 应用程序，用于与客户进行交互式用户会话。您的应用程序必须在用户的浏览器中调用此端点和其他托管登录端点。

要将您的用户重定向到托管登录以再次登录，请在请求中添加一个redirect_uri参数。带 redirect_uri 参数的 logout 请求还必须包含对登录端点的后续请求的参数，例如 client_id、response_type 和 scope。

要将用户重定向到您选择的页面，请在您的应用程序客户端中 URLs添加允许注销。在用户的对 logout 端点的请求中，添加 logout_uri 和 client_id 参数。如果的值logout_uri是您的应用程序客户端允许的注销 URLs之一，则 Amazon Cognito 会将用户重定向到该网址。

使用 SAML 2.0 的单点注销 (SLO)，Amazon IdPs Cognito 首先将您的用户重定向到您在 IdP 配置中定义的 SLO 终端节点。在您的 IdP 将您的用户重定向回到 saml2/logout 之后，Amazon Cognito 会根据您的请求，再向 redirect_uri 或 logout_uri 发送一次重定向响应。有关更多信息，请参阅通过单点注销来注销 SAML 用户。

注销端点不会让用户退出 OIDC 或社交身份提供商 ()。IdPs要让用户从与外部 IdP 进行的会话中注销，请将他们引导到该提供者的注销页面。

GET /logout

/logout 端点只支持 HTTPS GET。用户池客户端通常通过浏览器发出此请求。浏览器在 Android 中通常是自定义 Chrome 标签页，在 iOS 中是 Safari 视图控件。

请求参数

client_id

您的应用程序的应用程序客户端 ID。要获取应用程序客户端 ID，您必须在用户池中注册该应用程序。有关更多信息，请参阅 特定于应用程序的应用程序客户端设置。

必需。

logout_uri

使用 logout_uri 参数将用户重新导向到自定义注销页面。将其值设置为应用程序客户端注销 URL，您要在用户退出后将其重新导向到此 URL。仅将 logout_uri 与 client_id 参数一起使用。有关更多信息，请参阅 特定于应用程序的应用程序客户端设置。

您也可以使用 logout_uri 参数将用户重定向到另一个应用程序客户端的登录页面。将其他应用程序客户端的登录页面设置为您的应用程序客户端中的允许的回调 URL。在对 `/logout` 端点的请求中，将 logout_uri 参数的值设置为 URL 编码的登录页面。

Amazon Cognito 要求在您对 `/logout` 端点的请求中使用 logout_uri 或 redirect_uri 参数。logout_uri 参数会将您的用户重定向到另一个网站。如果您对 `/logout` 端点的请求中同时包含 logout_uri 和 redirect_uri 参数，Amazon Cognito 将仅使用 logout_uri 参数，不使用 redirect_uri 参数。

nonce

（可选）您可以添加到请求中的随机值。您提供的 nonce 值包含在 Amazon Cognito 发出的 ID 令牌中。为了防范重播攻击，您的应用程序可以检查 ID 令牌中的 `nonce` 声明并将其与您生成的声明进行比较。有关 `nonce` 声明的更多信息，请参阅《OpenID Connect 标准》中的 ID token validation（ID 令牌验证）。

redirect_uri

使用 redirect_uri 将用户重新导向到登录页以进行身份验证。将其值设置为应用程序客户端允许的回调 URL，您要在用户再次登录后将其重新导向到此 URL。添加您要传递给 `/login` 端点的 client_id、scope、state 和 response_type 参数。

Amazon Cognito 要求在您对 `/logout` 端点的请求中使用 logout_uri 或 redirect_uri 参数。要将用户重定向到 `/login` 端点以重新验证身份并将令牌传递给您的应用程序，请添加 redirect_uri 参数。如果您向 `/logout` 端点发出的请求中同时包含 logout_uri 和 redirect_uri 参数，则 Amazon Cognito 会覆盖 redirect_uri 参数，并以独占方式处理 logout_uri 参数。

response_type

您希望在用户登录后从 Amazon Cognito 收到的 OAuth 2.0 响应。 `code`和`token`是 r esponse_type 参数的有效值。

在您使用 redirect_uri 参数时是必需的。

状态

当应用程序向请求添加 state 参数时，如果 `/oauth2/logout` 端点重新导向您的用户，则 Amazon Cognito 将此参数的值返回给您的应用程序。

将此值添加到您的请求中以防止 CSRF 攻击。

不能将 `state` 参数的值设置为 URL 编码的 JSON 字符串。要在 `state` 参数中传递与此格式匹配的字符串，请将该字符串编码为 base64，然后在应用程序中对其进行解码。

如果您使用 redirect_uri 参数，强烈推荐使用此参数。

范围

你想在使用 redi re ct_uri 参数注销后向亚马逊 Cognito 请求的 OAuth 2.0 范围。Amazon Cognito 使用您对 `/logout` 端点的请求中的 scope 参数将您的用户重定向到 `/login` 端点。

在您使用 redirect_uri 参数时是可选的。如果不包括 scope 参数，Amazon Cognito 会使用 scope 参数将您的用户重定向到 `/login` 端点。当 Amazon Cognito 重定向用户并自动填充 `scope` 时，该参数包括应用程序客户端的所有授权范围。

示例请求

示例：注销并将用户重新导向到客户端

当请求包含 `logout_uri` 和 `client_id` 时，Amazon Cognito 会将用户会话重定向到 `logout_uri` 值中的 URL，忽略所有其他请求参数。这个 URL 必须是应用程序客户端的授权注销 URL。

以下是注销并重定向到 `https://www.example.com/welcome` 的请求示例。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  client_id=1example23456789&
  logout_uri=https%3A%2F%2Fwww.example.com%2Fwelcome
```

示例：注销并提示用户以其他用户身份登录

当请求省略logout_uri但以其他方式提供了构成向授权终端节点发出的格式良好的请求的参数时，Amazon Cognito 会将用户重定向到托管登录登录。注销端点会将原始请求中的参数附加到重定向目的地。

您添加到注销请求中的其他参数必须位于的列表中。请求参数例如，注销端点不支持identity_provider使用idp_identifier或参数的自动 IdP 重定向。向注销终端节点发出的请求redirect_uri中的参数不是注销 URL，而是您要传递到授权端点的 post-sign-in URL。

以下是将用户注销，重定向到登录页面，然后在登录后向 https://www.example.com 提供授权代码的请求示例。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/logout?
  response_type=code&
  client_id=1example23456789&
  redirect_uri=https%3A%2F%2Fwww.example.com&
  state=example-state-value&
  nonce=example-nonce-value&
  scope=openid+profile+aws.cognito.signin.user.admin
```

身份提供者和依赖方端点

联合身份验证端点是用户池端点，用于支持用户池使用的其中一个身份验证标准。它们包括 SAML ACS URLs、OIDC 发现端点以及作为身份提供者和信赖方的用户池角色的服务端点。联合端点启动身份验证流程，接收来自客户端的身份验证证明 IdPs，并向客户端发放令牌。他们与 IdPs应用程序和管理员交互，但不与用户交互。

本页之后的整页主题包含有关 OAuth 2.0 和 OIDC 提供商端点的详细信息，当您向用户池中添加域名时，这些终端节点将变为可用。下图是所有联合身份验证端点的列表。

用户池域的示例有：

1. 前缀域：mydomain.auth.us-east-1.amazoncognito.com
2. 自定义域名：auth.example.com

用户池联合身份验证端点

端点 URL	描述	如何访问此端点
https://*Your user pool domain*/oauth2/授权	将用户重定向到托管登录或使用其 IdP 登录。	在客户浏览器中调用以开始用户身份验证。请参阅对端点授权。
https:// *Your user pool domain* /oauth2/tok	根据授权码或客户端凭证请求返回令牌。	应用程序请求检索令牌。请参阅令牌端点。
https://*Your user pool domain*/oauth2/userInfo	根据 OAuth 2.0 范围和访问令牌中的用户身份返回用户属性。	应用程序请求检索用户配置文件。请参阅userInfo 端点。
https://*Your user pool domain*/oauth2/revoke	撤消刷新令牌和关联的访问令牌。	应用程序请求撤销令牌。请参阅撤消端点。
https://cognito-idp。*Region*.amazonaws.com/ /.wellknown/openid 配置 *your user pool ID*	用户池的 OIDC 架构的目录。	应用程序请求查找用户池发布者元数据。
https://cognito-idp。*Region*.amazonaws.com/ /.well *your user pool ID*-known/jwks.json	您可以用来验证 Amazon Cognito 令牌的公有密钥。	由应用程序请求进行验证 JWTs。
https:/*Your user pool domain*/oauth2/idpresponse	社交身份提供者必须使用授权码将用户重新导向到此端点。Amazon Cognito 在验证您的联合用户时将代码兑换为令牌。	已从 OIDC IdP 登录重定向为 IdP 客户端回调 URL。
https://*Your user pool domain*/saml2/idpresponse	与 SAML 2.0 身份提供者集成所需的断言使用者响应（ACS）URL。	已从 SAML 2.0 IdP 重定向为 ACS URL，或 IdP 发起的登录的起点[1]。

端点 URL	描述	如何访问此端点
https://*Your user pool domain*/saml2/logout	用于与 SAML 2.0 身份提供者集成的单点注销（SLO）URL。	已从 SAML 2.0 IdP 重定向为单点注销（SLO）URL。仅接受 POST 绑定。

[1] 有关 IdP 发起的 SAML 登录的更多信息，请参阅 实施 IDP 发起的 SAML 登录。

有关 OpenID Connect 和 OAuth 标准的更多信息，请参阅 OpenID Con nect 1.0 和 2.0。OAuth

主题

- 重定向和授权端点
- 令牌发布者端点
- 用户属性端点
- 令牌撤销端点
- IdP SAML 断言端点

重定向和授权端点

`/oauth2/authorize` 端点是支持两个重定向目标的重定向端点。如果您在 URL 中包括 `identity_provider` 或 `idp_identifier` 参数，则它会静默地将用户重定向到该身份提供者（IdP）的登录页面。否则，它会使用您在请求中包括的相同 URL 参数重定向到 登录端点。

授权端点重定向到托管登录或 IdP 登录页面。此端点上的用户会话的目的地是您的用户必须直接在其浏览器中与之交互的网页。

要使用 authorize 端点，请在 `/oauth2/authorize` 上使用为您的用户池提供有关以下用户池详细信息的信息的参数调用用户的浏览器。

- 您希望登录到的应用程序客户端。
- 您希望最终到达的回调 URL。
- 您要在用户的访问令牌中请求的 OAuth 2.0 范围。
- （可选）您希望用于登录的第三方 IdP。

您还可以提供 Amazon Cognito 用来验证传入声明的 `state` 和 `nonce` 参数。

GET **/oauth2/authorize**

`/oauth2/authorize` 端点只支持 `HTTPS GET`。您的应用程序通常在用户的浏览器中发起此请求。您只能通过 HTTPS 向 `/oauth2/authorize` 端点发出请求。

您可以在 OpenID Connect（OIDC）标准的授权端点中详细了解授权端点的定义。

请求参数

`response_type`

必需。

响应类型。必须为 `code` 或 `token`。

`response_type` 为 `code` 的成功请求返回授权代码授予。授权代码授予是 Amazon Cognito 附加到重定向 URL 的 `code` 参数。您的应用程序可以将包含 令牌端点的代码交换为访问权限、ID 和刷新令牌。作为安全最佳实践，以及要为您的用户接收刷新令牌，请在您的应用程序中使用授权代码授予。

`response_type` 为 `token` 的成功请求返回隐式授予。隐式授予是 Amazon Cognito 附加到您的重定向 URL 的 ID 和访问令牌。隐式授予的安全性较差，因为它会向用户公开令牌和潜在的识别信息。您可以在应用程序客户端的配置中停用对隐式授予的支持。

`client_id`

必需。

应用程序客户端 ID。

`client_id` 的值必须是您在其中发出请求的用户池中应用程序客户端的 ID。您的应用程序客户端必须支持由 Amazon Cognito 本地用户或至少一个第三方 IdP 登录。

`redirect_uri`

必需。

在 Amazon Cognito 授权用户之后，身份验证服务器将浏览器重定向到的 URL。

重定向统一资源标识符（URI）必须具有以下属性：

- 必须是绝对 URI。
- 您必须已经将 URI 预注册到客户端。

- 不能包含片段组件。

请参阅 OAuth 2.0-重定向端点。

Amazon Cognito 要求您的重新导向 URI 使用 HTTPS，但 `http://localhost` 除外，您可以将其设置为回调 URL 以进行测试。

Amazon Cognito 还支持应用程序回调， URLs 例如。`myapp://example`

`state`

可选，推荐。

当应用程序向请求添加 state 参数时，如果 `/oauth2/authorize` 端点重新导向您的用户，则 Amazon Cognito 将此参数的值返回给您的应用程序。

将此值添加到您的请求中以防止 CSRF 攻击。

不能将 `state` 参数的值设置为 URL 编码的 JSON 字符串。要在 `state` 参数中传递与此格式匹配的字符串，请将该字符串编码为 base64，然后在应用程序中对其进行解码。

`identity_provider`

可选。

添加此参数可绕过托管登录并将您的用户重定向到提供商登录页面。identity_provider 参数的值是出现在您的用户池中的身份提供商（IdP）的名称。

- 对于社交提供者，您可以使用 identity_provider 值 `Facebook`、`Google`、`LoginWithAmazon` 和 `SignInWithApple`。
- 对于 Amazon Cognito 用户池，使用值 `COGNITO`。
- 对于 SAML 2.0 和 OpenID Connect (OIDC) 身份提供商 (IdPs)，请使用您在用户池中分配给 IdP 的名称。

`idp_identifier`

可选。

添加此参数以重定向到具有 identity_provider 名称的替代名称的提供程序。您可以 IdPs 从 Amazon Cognito 控制台的社交和外部提供商菜单中输入 SAML 2.0 和 OIDC 的标识符。

`scope`

可选。

可以是任何系统预留范围或与客户端关联的自定义范围的组合。范围必须以空格分隔。系统预留范围为 `openid`、`email`、`phone`、`profile` 和 `aws.cognito.signin.user.admin`。使用的任意范围必须与客户端关联，否则将在运行时忽略。

如果客户端不请求任何范围，则身份验证服务器使用与客户端关联的所有范围。

如果请求 `openid` 范围，则只返回 ID 令牌。如果请求 `aws.cognito.signin.user.admin` 范围，则访问令牌只能用于 Amazon Cognito 用户池。如果同时请求了 phone 范围，则只能请求 `email`、`profile` 和 `openid` 范围。这些范围控制进入 ID 令牌中的声明。

`code_challenge_method`

可选。

您用来生成质询的哈希协议。PKCE RFC 定义两个方法：S256 和 plain；但是，Amazon Cognito 身份验证服务器仅支持 S256。

`code_challenge`

可选。

您从中生成的密钥代码交换证明 (PKCE) 质询。`code_verifier`有关更多信息，请参阅 在授权代码授予中使用 PKCE。

仅当您指定 `code_challenge_method` 参数时是必需的。

`nonce`

可选。

您可以添加到请求中的随机值。您提供的 nonce 值包含在 Amazon Cognito 发出的 ID 令牌中。为了防范重播攻击，您的应用程序可以检查 ID 令牌中的 `nonce` 声明并将其与您生成的声明进行比较。有关 `nonce` 声明的更多信息，请参阅《OpenID Connect 标准》中的 ID token validation（ID 令牌验证）。

`lang`

可选。

您想要显示用户交互式页面的语言。托管登录页面可以本地化，但托管用户界面（经典）页面不能。有关更多信息，请参阅 托管登录本地化。

`login_hint`

可选。

要传递给授权服务器的用户名提示。您可以从用户那里收集用户名、电子邮件地址或电话号码，并允许目的地提供者预先填入用户的登录名。当您向`oauth2/authorize`终端节点提交`login_hint`参数和否`idp_identifier`或`identity_provider`参数时，托管登录会使用您的提示值填充用户名字段。您也可以将此参数传递给 登录端点 并自动填入用户名值。

当您的授权请求调用指向 OIDC IdPs 或 Google 的重定向时，Amazon Cognito 会向该第三方授权机构在请求中添加一个`login_hint`参数。你无法将登录提示转发给 SAML、Apple、Login With Amazon 或 Facebook（Meta） IdPs。

`prompt`

可选。

一个 OIDC 参数，用于控制现有会话的身份验证行为。仅在托管登录品牌版本中可用，不适用于经典托管用户界面。有关 OIDC 规范的更多信息，请参阅身份验证请求。值`none`和`login`会影响用户池的身份验证行为。

IdPs 当用户选择向第三方提供商进行身份验证时，Amazon Cognito 会`none`将`prompt`除您的值之外的所有值转发给您。当用户访问的 URL 包含`identity_provider`或`idp_identifier`参数，或者授权服务器将他们重定向到，然后他们从可用按钮中选择登录端点和 IdP 时，就会出现这种情况。

提示参数值

`prompt=none`

Amazon Cognito 会以静默方式继续对拥有有效身份验证会话的用户进行身份验证。使用此提示，用户可以在用户池中的不同应用程序客户端之间进行静默身份验证。如果用户尚未通过身份验证，则授权服务器会返回`login_required`错误。

`prompt=login`

Amazon Cognito 要求用户重新进行身份验证，即使他们已有会话。当您想再次验证用户的身份时，请发送此值。拥有现有会话的经过身份验证的用户可以在不使该会话失效的情况下返回登录状态。当已有会话的用户再次登录时，Amazon Cognito 会为他们分配一个新的会话 Cookie。也可以将此参数转发给您的 IdPs。 IdPs接受此参数的用户还会向用户请求新的身份验证尝试。

`prompt=select_account`

此值对本地登录没有影响，必须在重定向到 IdPs的请求中提交。当包含在您的授权请求中时，此参数将添加`prompt=select_account`到 IdP 重定向目标的 URL 路径中。当 IdPs 支持此参数时，他们会要求用户选择他们想要登录的账户。

prompt=consent

此值对本地登录没有影响，必须在重定向到 IdPs的请求中提交。当包含在您的授权请求中时，此参数将添加prompt=consent到 IdP 重定向目标的 URL 路径中。当 IdPs支持此参数时，他们会先征得用户同意，然后再重定向回您的用户池。

当您在请求中省略该prompt参数时，托管登录将遵循默认行为：除非他们的浏览器具有有效的托管登录会话 Cookie，否则用户必须登录。例如，您可以将的多个值prompt与空格字符分隔符组合在一起。prompt=login consent

示例：授权码授权

这是授权代码授予的示例请求。

以下请求会启动会话，以便检索您的用户在 redirect_uri 目的地传递给应用程序的授权代码。该会话请求的范围包括用户属性和对 Amazon Cognito 自助服务 API 操作的访问权限。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin
```

Amazon Cognito 身份验证服务器使用授权代码和状态重新导向回您的应用程序。授权代码的有效期为五分钟。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

示例：使用 PKCE 授予授权码

此示例流程使用 PKCE 执行授权码授权。

此请求添加了一个code_challenge参数。要完成令牌代码的交换，必须在向 /oauth2/token 端点发出的请求中包括 code_verifier 参数。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
```

```
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin&
code_challenge_method=S256&
code_challenge=a1b2c3d4...
```

授权服务器使用授权码和状态重定向回您的应用程序。您的应用程序处理授权码并将其兑换为令牌。

```
HTTP/1.1 302 Found
Location: https://www.example.com?code=a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111&state=abcdefg
```

示例：要求使用重新进行身份验证 **prompt=login**

以下请求添加了一个prompt=login参数，要求用户再次进行身份验证，即使他们已有会话。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin&
prompt=login
```

授权服务器重定向到登录端点，需要重新进行身份验证。

```
HTTP/1.1 302 Found Location: https://mydomain.auth.us-east-1.amazoncognito.com/
login?response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&state=abcdefg&scope=openid+profile
+aws.cognito.signin.user.admin&prompt=login
```

示例：使用静默身份验证 **prompt=none**

以下请求添加了一个prompt=none参数，用于静默检查用户是否有有效的会话。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=openid+profile+aws.cognito.signin.user.admin&
prompt=none
```

当不存在有效的会话时，授权服务器会向重定向 URI 返回错误

```
HTTP/1.1 302 Found Location: https://www.example.com?error=login_required&state=abcdefg
```

存在有效会话时，授权服务器会返回授权码。

```
HTTP/1.1 302 Found Location: https://www.example.com?
code=AUTHORIZATION_CODE&state=abcdefg
```

示例：没有**openid**作用域的代币（隐式）授权

此示例流程生成隐式授权，并 JWTs 直接返回到用户的会话。

请求来自您的授权服务器的隐式授权。它请求访问令牌中的作用域来授权用户配置文件自助服务操作。

```
GET https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
scope=aws.cognito.signin.user.admin
```

授权服务器仅使用访问令牌重定向回您的应用程序。由于未请求 `openid` 范围，Amazon Cognito 不会返回 ID 令牌。此外，Amazon Cognito 不会在此流程中返回刷新令牌。

```
HTTP/1.1 302 Found
Location: https://example.com/
callback#access_token=eyJra456defEXAMPLE&token_type=bearer&expires_in=3600&state=STATE
```

示例：带**openid**作用域的代币（隐式）授权

此示例流程生成隐式授权，并将令牌返回给用户的浏览器。

请求来自您的授权服务器的隐式授权。它请求访问令牌中的作用域，以授权访问用户属性和自助服务操作。

```
GET
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=token&
client_id=1example23456789&
redirect_uri=https://www.example.com&
state=abcdefg&
```

```
scope=aws.cognito.signin.user.admin+openid+profile
```

授权服务器使用访问令牌和 ID 令牌重定向回您的应用程序（因为包含了openid范围）：

```
HTTP/1.1 302 Found
Location: https://
www.example.com#id_token=eyJra67890EXAMPLE&access_token=eyJra12345EXAMPLE&token_type=bearer&exp
```

负向响应的示例

Amazon Cognito 可能会拒绝您的请求。失败的请求附带 HTTP 错误代码和相关描述，您可以使用这些信息来更正请求参数。以下是负向响应的示例。

- 如果 `client_id` 和 `redirect_uri` 有效，但请求参数格式不正确，身份验证服务器会将该错误重定向到客户端的 `redirect_uri` 并在 URL 参数中附加错误消息。以下是错误格式的示例。
  - 该请求不包括 `response_type` 参数。
  - 授权请求提供了 `code_challenge` 参数，但没有提供 `code_challenge_method` 参数。
  - `code_challenge_method` 参数的值不是 `S256`。

  以下是使用了错误格式的示例请求的响应。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_request
  ```

- 如果客户端在 `response_type` 中请求 `code` 或 `token`，但没有这些请求的权限，则 Amazon Cognito 授权服务器将 `unauthorized_client` 返回到客户端的 `redirect_uri`，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=unauthorized_client
  ```

- 如果客户端请求范围未知、格式错误或者无效，则 Amazon Cognito 授权服务器会将 `invalid_scope` 返回到客户端的 `redirect_uri`，如下所示：

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=invalid_scope
  ```

- 如果服务器中有意外的错误，则授权服务器会将 `server_error` 返回到客户端的 `redirect_uri`。由于 HTTP 500 错误不会发送到客户端，所以在用户的浏览器中不会显示该错误。授权服务器返回以下错误。

  ```
  HTTP 1.1 302 Found Location: https://client_redirect_uri?error=server_error
  ```

- 当 Amazon Cognito 通过联合身份验证向第三方 IdPs进行身份验证时，Amazon Cognito 可能会遇到连接问题，例如：
  - 如果从 IdP 处请求令牌时连接超时，身份验证服务器会将该错误重定向到客户端的 `redirect_uri`，如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Timeout+occurred+in+calling+IdP+token
    +endpoint
    ```

  - 如果在调用 `jwks_uri` 端点进行 ID 令牌验证时连接超时，身份验证服务器会重定向到客户端的 `redirect_uri`，并出现如下所示的错误：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=error_description=Timeout+in+calling+jwks
    +uri
    ```

- 通过联合第三方进行身份验证时 IdPs，提供商可能会返回错误响应。这可能是由于配置错误或其他原因而造成，例如以下原因：
  - 如果从其他提供商处收到错误响应，身份验证服务器会将该错误重定向到客户端的 `redirect_uri`，如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=[IdP name]+Error+-+[status code]+error
     getting token
    ```

  - 如果从 Google 收到错误响应，身份验证服务器会将该错误重定向到客户端的 `redirect_uri`，如下所示：

    ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Google+Error+-+[status code]+[Google-
    provided error code]
    ```

- 如果 Amazon Cognito 在连接到外部 IdP 时遇到通信异常，则身份验证服务器会重定向到客户端的 `redirect_uri`，并显示以下错误消息：
  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Connection+reset
    ```

  - ```
    HTTP 1.1 302 Found Location: https://client_redirect_uri?
    error=invalid_request&error_description=Read+timed+out
    ```

令牌发布者端点

的 OAuth 2.0 令牌端点向想要完成授权码和客户端凭证授予流程的应用程序`/oauth2/token`发放 JSON Web 令牌 (JWTs)。这些令牌是使用用户池进行身份验证的最终结果。这些令牌包含有关用户（ID 令牌）、用户的访问级别（访问令牌）以及用户保留其已登录会话的权利（刷新令牌）的信息。OpenID Connect（OIDC）依赖方库处理向此端点发出的请求以及来自此端点的响应有效载荷。令牌提供可验证的身份验证证明、配置文件信息以及用于访问后端系统的机制。

您的用户池 OAuth 2.0 授权服务器从令牌端点向以下类型的会话发放 JSON Web 令牌 (JWTs)：

1. 已完成授权码授权请求的用户。成功兑换代码会返回 ID、访问令牌和刷新令牌。
2. Machine-to-machine (M2M) 已完成客户端凭证授予的会话。成功使用客户端机密进行授权会返回访问令牌。
3. 之前已登录并收到刷新令牌的用户。刷新令牌身份验证会返回新的 ID 令牌和访问令牌。

> **Note**
>
> 使用授权码进行托管登录或通过联合身份登录登录的用户可以随时从令牌端点刷新其令牌。当记忆设备在您的用户池中未处于活动状态时，使用 API 操作 `InitiateAuth` 和 `AdminInitiateAuth` 登录的用户可以使用令牌端点刷新其令牌。如果记住的设备处于活动状态，请使用应用客户端的相关 API 或 SDK 令牌刷新操作刷新令牌。

当您向用户池添加域时，令牌端点将公开可用。它接受 HTTP POST 请求。为了实现应用程序安全性，请将 PKCE 与授权代码登录事件结合使用。PKCE 会验证传递授权码的用户是否与经过身份验证的用户相同。有关 PKCE 的更多信息，请参阅 IETF RFC 7636。

要详细了解用户池应用程序客户端及其授权类型、客户端密钥、允许的范围和客户端，请 IDs 访问特定于应用程序的应用程序客户端设置。要详细了解 M2M 授权、客户端凭证授权以及使用访问令牌范围的授权，请访问作用域、M2M 以及 APIs 带资源服务器的场景。

要从用户的访问令牌中检索有关用户的信息，可以将其传递给 userInfo 端点 或 GetUser API 请求。访问令牌必须包含这些请求的相应范围，

格式化对令牌端点的 POST 请求

`/oauth2/token` 端点只支持 `HTTPS POST`。此端点不是用户交互式的。使用应用程序中的 OpenID Connect (OIDC) 库处理令牌请求。

令牌端点支持 client_secret_basic 和 client_secret_post 身份验证。有关 OIDC 规范的更多信息，请参阅客户端身份验证。有关 OpenID Connect 规范中令牌端点的更多信息，请参阅令牌端点。

标头中的请求参数

您可以将请求标头中的以下参数传递给令牌端点。

Authorization

如果向客户端发布了密钥，则客户端可以在授权标头中将其 client_id 和 client_secret 作为 client_secret_basic HTTP 授权传递。您还可以在请求正文中包含 client_id 和 client_secret 作为 client_secret_post 授权。

授权标头字符串是 基本 Base64Encode(client_id:client_secret)。以下示例是具有客户端密钥 abcdef01234567890 的应用程序客户端 djc98u3jiedmi283eu928 的授权标头，其使用字符串 djc98u3jiedmi283eu928:abcdef01234567890 的 Base64 编码版本：

```
Authorization: Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw
```

Content-Type

将此参数的值设置为 'application/x-www-form-urlencoded'。

正文中的请求参数

以下是您可以在请求正文中以x-www-form-urlencoded格式向令牌端点请求的参数。

grant_type

必填项。

您要申请的 OIDC 拨款类型。

必须为 authorization_code、refresh_token 或 client_credentials。在以下条件下，您可以从令牌端点请求自定义范围的访问令牌：

- 您在应用程序客户端配置中启用请求的范围。
- 您为应用程序客户端配置了客户端密钥。
- 您在应用程序客户端中启用客户端凭证授予。

> **Note**
>
> 仅当grant_type是时，令牌端点才会返回刷新令牌authorization_code。

client_id

可选。当您在*Authorization*标题中提供应用程序客户端 ID 时，这不是必填项。

用户池中应用程序客户端的 ID。指定对用户进行身份验证的同一个应用程序客户端。

如果客户端为公有且没有密钥，或在 client_secret_post 授权中有 client_secret，则必须提供此参数。

client_secret

可选。当您在*Authorization*标题中提供客户端密钥以及应用程序客户端没有密钥时，则无需填写。

用于client_secret_post授权的应用程序客户端密钥（如果应用程序客户端有）。

scope

可选。

可以是与您的应用程序客户端关联的任何范围的组合。Amazon Cognito 会忽略请求中不允许被请求的应用程序客户端使用的范围。如果您不提供此请求参数，则授权服务器会返回访问令牌scope声明，其中包含您在应用程序客户端配置中启用的所有授权范围。您可以请求所请求的应用程序客户端允许的任何范围：标准范围、来自资源服务器的自定义作用域以及aws.cognito.signin.user.admin用户自助服务范围。

redirect_uri

可选。授予客户凭证不是必需的。

必须是用于在 /oauth2/authorize 中获取 authorization_code 的相同 redirect_uri。

如果 grant_type 是 authorization_code，则必须提供此参数。

refresh_token

可选。仅当用户已经拥有刷新令牌并希望获得新 ID 和访问令牌时使用。

要为用户的会话生成新的访问权限和 ID 令牌，请将的值设置refresh_token为请求的应用程序客户端发放的有效刷新令牌。

当刷新令牌轮换处于活动状态时，返回带有新 ID 和访问令牌的新刷新令牌，否则仅返回 ID 和访问令牌。

code

可选。仅在授权码授权中才需要。

授权码授予的授权码。如果授权请求包括 `authorization_code` 的 `grant_type`，则必须提供此参数。

aws_client_metadata

可选。

您要传递给的信息令牌生成前 Lambda 触发器。您的应用程序可以收集有关用户或计算机会话的上下文信息，并将其传递到此参数中。当您以 URL 编码的 JSON 格式传递`aws_client_metadata`时，Amazon Cognito 会将其包含在触发器 Lambda 函数的输入事件中。您的预令牌触发事件版本或全局 Lambda 触发器版本必须配置为版本二或更高版本。

code_verifier

可选。仅当您在初始授权请求中提供了`code_challenge_method`和`code_challenge`参数时才为必填项。

您的应用程序在 PK CE 的授权码授予请求`code_challenge`中计算的生成的代码验证器。

为获取令牌交换授权代码

使用授权码授权进行身份验证后，以下请求成功生成 ID、访问和刷新令牌。请求在`Authorization`标头中以`client_secret_basic`格式传递客户机密钥。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token&
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
code=AUTHORIZATION_CODE&
redirect_uri=com.myclientapp://myclient/redirect
```

响应会向用户发放新的 ID、访问和刷新令牌以及其他元数据。

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "refresh_token": "eyJj3example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

具有基本授权的客户凭证

以下来自 M2M 应用程序的请求请求授予客户端证书。由于客户端凭据需要客户端密钥，因此使用从应用程序客户端 ID 和密钥派生的`Authorization`标头对请求进行授权。该请求会生成一个具有两个请求范围的访问令牌。该请求还包括提供 IP 地址信息的客户端元数据以及向该授权所代表的用户颁发的令牌。Amazon Cognito 将客户端元数据传递给令牌生成前 Lambda 触发器。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=client_credentials&
client_id=1example23456789&
scope=resourceServerIdentifier1%2Fscope1%20resourceServerIdentifier2%2Fscope2&
&aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,
%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito 将以下输入事件传递给代币生成前 Lambda 触发器。

```
{
    version: '3',
    triggerSource: 'TokenGeneration_ClientCredentials',
    region: 'us-east-1',
    userPoolId: 'us-east-1_EXAMPLE',
    userName: 'ClientCredentials',
    callerContext: {
        awsSdkVersion: 'aws-sdk-unknown-unknown',
        clientId: '1example23456789'
    },
    request: {
        userAttributes: {},
        groupConfiguration: null,
```

```
        scopes: [
           'resourceServerIdentifier1/scope1',
           'resourceServerIdentifier2/scope2'
        ],
        clientMetadata: {
            'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
            'ClientIpAddress': '192.0.2.252'
        }
    },
    response: { claimsAndScopeOverrideDetails: null }
}
```

响应返回访问令牌。客户凭证授予用于 machine-to-machine（M2M）授权，并且仅返回访问令牌。

```
HTTP/1.1 200 OK
Content-Type: application/json
{
    "access_token": "eyJra1example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

具有 POST 正文授权的客户端凭证

以下客户端凭证授予请求在请求正文中包含该`client_secret`参数，并且不包含`Authorization`标头。此请求使用`client_secret_post`授权语法。该请求会生成一个具有请求范围的访问令牌。该请求还包括提供 IP 地址信息的客户端元数据以及向该授权所代表的用户颁发的令牌。Amazon Cognito 将客户端元数据传递给令牌生成前 Lambda 触发器。

```
POST /oauth2/token HTTP/1.1
Content-Type: application/x-www-form-urlencoded
X-Amz-Target: AWSCognitoIdentityProviderService.Client credentials request
User-Agent: USER_AGENT
Accept: /
Accept-Encoding: gzip, deflate, br
Content-Length: 177
Referer: http://auth.example.com/oauth2/token
Host: auth.example.com
Connection: keep-alive

grant_type=client_credentials&
```

```
client_id=1example23456789&
scope=my_resource_server_identifier%2Fmy_custom_scope&
client_secret=9example87654321&
aws_client_metadata=%7B%22onBehalfOfToken%22%3A%22eyJra789ghiEXAMPLE%22,
%20%22ClientIpAddress%22%3A%22192.0.2.252%22%7D
```

Amazon Cognito 将以下输入事件传递给代币生成前 Lambda 触发器。

```
{
    version: '3',
    triggerSource: 'TokenGeneration_ClientCredentials',
    region: 'us-east-1',
    userPoolId: 'us-east-1_EXAMPLE',
    userName: 'ClientCredentials',
    callerContext: {
        awsSdkVersion: 'aws-sdk-unknown-unknown',
        clientId: '1example23456789'
    },
    request: {
        userAttributes: {},
        groupConfiguration: null,
        scopes: [
           'resourceServerIdentifier1/my_custom_scope'
        ],
        clientMetadata: {
            'onBehalfOfToken': 'eyJra789ghiEXAMPLE',
            'ClientIpAddress': '192.0.2.252'
        }
    },
    response: { claimsAndScopeOverrideDetails: null }
}
```

响应返回访问令牌。客户凭证授予用于 machine-to-machine（M2M）授权，并且仅返回访问令牌。

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Date: Tue, 05 Dec 2023 16:11:11 GMT
x-amz-cognito-request-id: 829f4fe2-a1ee-476e-b834-5cd85c03373b

{
    "access_token": "eyJra12345EXAMPLE",
    "expires_in": 3600,
    "token_type": "Bearer"
```

```
}
```

具有 PKCE 的授权代码授予

以下示例请求完成了授权请求，该请求包含在 PKCE 的授权码授予请求中包含code_challenge_method和code_challenge参数。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=authorization_code&
client_id=1example23456789&
code=AUTHORIZATION_CODE&
code_verifier=CODE_VERIFIER&
redirect_uri=com.myclientapp://myclient/redirect
```

响应返回应用程序成功验证 PKCE 的 ID、访问和刷新令牌。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "refresh_token": "eyJj3example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

无需刷新令牌轮换即可刷新令牌

以下示例请求向刷新令牌轮换处于非活动状态的应用程序客户端提供刷新令牌。由于应用程序客户端有客户端密钥，因此请求会提供标Authorization头。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
refresh_token=eyJj3example
```

响应会返回新的 ID 和访问令牌。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

使用刷新令牌轮换刷新令牌

以下示例请求向启用刷新令牌轮换的应用程序客户端提供刷新令牌。由于应用程序客户端有客户端密钥，因此请求会提供标Authorization头。

```
POST https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/token >
Content-Type='application/x-www-form-urlencoded'&
Authorization=Basic ZGpjOTh1M2ppZWRtaTI4M2V1OTI4OmFiY2RlZjAxMjM0NTY3ODkw

grant_type=refresh_token&
client_id=1example23456789&
refresh_token=eyJj3example
```

响应会返回新的 ID、访问和刷新令牌。

```
HTTP/1.1 200 OK
Content-Type: application/json

{
    "access_token": "eyJra1example",
    "id_token": "eyJra2example",
    "refresh_token": "eyJj4example",
    "token_type": "Bearer",
    "expires_in": 3600
}
```

负向响应的示例

格式错误的请求会从令牌端点生成错误。以下是令牌请求生成错误时响应正文的一般映射。

```
HTTP/1.1 400 Bad Request
Content-Type: application/json;charset=UTF-8

{
"error":"invalid_request|invalid_client|invalid_grant|unauthorized_client|
unsupported_grant_type"
}
```

invalid_request

请求缺少必需的参数、包括不支持的参数值（除了 unsupported_grant_type 之外）或者格式错误。例如，grant_type 是 refresh_token，但未包括 refresh_token。

invalid_client

客户端身份验证失败。例如，客户端的授权标头中包含 client_id 和 client_secret，但没有这样的客户端带有 client_id 和 client_secret。

invalid_grant

已撤销刷新令牌。

授权代码已使用或不存在。

应用程序客户端对请求的范围内的所有属性都没有读取权限。例如，您的应用程序请求 email 范围，应用程序客户端可以读取 email 属性，但不能读取 email_verified。

unauthorized_client

客户端不允许代码授予流或刷新令牌。

unsupported_grant_type

如果 grant_type 是 authorization_code、refresh_token 或 client_credentials 之外的任意内容，则返回。

用户属性端点

在 OIDC 发布包含用户属性的 ID 令牌的地方， OAuth 2.0 实现终端节点。/oauth2/userInfo经过身份验证的用户或客户端会收到包含 scopes 声明的访问令牌。此声明决定了授权服务器应返回的属性。当应用程序向 userInfo 端点提供访问令牌时，授权服务器会返回一个响应正文，其中包含通过访问令牌范围设置的边界内的用户属性。只要您的应用程序持有至少具有 openid 范围声明的有效访问令牌，就可以从 userInfo 端点检索有关用户的信息。

`userInfo` 端点是 OpenID Connect（OIDC）userInfo 端点。当服务提供商出示您的令牌端点发放的访问令牌时，响应中会包含用户属性。用户访问令牌中的范围定义了 userInfo 端点在其响应中返回的用户属性。`openid` 范围必须是访问令牌声明的范围之一。

Amazon Cognito 颁发访问令牌来响应用户池 API 请求，如 InitiateAuth。由于它们不包含任何范围，因此 userInfo 端点不接受这些访问令牌。而是必须由您出示来自令牌端点的访问令牌。

您的 OAuth 2.0 第三方身份提供商 (IdP) 还托管一个userInfo终端节点。当您的用户使用该 IdP 进行身份验证时，Amazon Cognito 会以静默方式与 IdP `token` 端点交换授权代码。您的用户池传递 IdP 访问令牌以授权从 IdP `userInfo` 端点检索用户信息。

用户访问令牌中的范围由身份验证`scopes`请求中的请求参数或令牌生成前 Lambda 触发器添加的范围决定。您可以解码访问令牌并检查`scope`声明，以查看其包含的访问控制范围。以下是一些影响从`userInfo`端点返回的数据的范围组合。预留的 Amazon Cognito 范围`aws.cognito.signin.user.admin`对从该终端节点返回的数据没有影响。

访问令牌中的示例作用域及其对**`userInfo`**响应的影响

`openid`

返回包含应用程序客户端可以读取的所有用户属性的响应。

`openid profile`

返回用户属性`name`、、`family_name`、`given_name`、`middle_name`、`nickname`、`preferred_username`、`pr`和`updated_at`。还会返回自定义属性。在没有每个属性的读取权限的应用程序客户端中，对此范围的响应是您的应用程序客户端拥有读取权限的规范中的所有属性。

`openid email`

返回基本配置文件信息以及`email`和`email_verified`属性。

`openid phone`

返回基本配置文件信息以及`phone_number`和`phone_number_verified`属性。

GET /oauth2/userInfo

您的应用程序直接生成对该端点的请求，而不是通过浏览器。

有关更多信息，请参阅 OpenID Connect（OIDC）规范中的 UserInfo 端点。

主题

- 标头中的请求参数
- 示例：请求
- 示例：正向响应
- 示例：负向响应

标头中的请求参数

Authorization: Bearer *<access_token>*

传递授权标头字段中的访问令牌。

必需。

示例：请求

```
GET /oauth2/userInfo HTTP/1.1
Content-Type: application/x-amz-json-1.1
Authorization: Bearer eyJra12345EXAMPLE
User-Agent: [User agent]
Accept: */*
Host: auth.example.com
Accept-Encoding: gzip, deflate, br
Connection: keep-alive
```

示例：正向响应

```
HTTP/1.1 200 OK
Content-Type: application/json;charset=UTF-8
Content-Length: [Integer]
Date: [Timestamp]
x-amz-cognito-request-id: [UUID]
X-Content-Type-Options: nosniff
X-XSS-Protection: 1; mode=block
Cache-Control: no-cache, no-store, max-age=0, must-revalidate
Pragma: no-cache
Expires: 0
Strict-Transport-Security: max-age=31536000 ; includeSubDomains
X-Frame-Options: DENY
Server: Server
```

```
Connection: keep-alive
{
    "sub": "[UUID]",
    "email_verified": "true",
    "custom:mycustom1": "CustomValue",
    "phone_number_verified": "true",
    "phone_number": "+12065551212",
    "email": "bob@example.com",
    "username": "bob"
}
```

有关 OIDC 声明的列表，请参阅标准声明。目前，Amazon Cognito 将 `email_verified` 和 `phone_number_verified` 的值返回为字符串。

示例：负向响应

示例：不正确的请求

```
HTTP/1.1 400 Bad Request
WWW-Authenticate: error="invalid_request",
error_description="Bad OAuth2 request at UserInfo Endpoint"
```

`invalid_request`

请求缺少必需的参数、包括不支持的参数值或格式错误。

示例：不正确的令牌

```
HTTP/1.1 401 Unauthorized
WWW-Authenticate: error="invalid_token",
error_description="Access token is expired, disabled, or deleted, or the user has
 globally signed out."
```

`invalid_token`

访问令牌已过期、已撤销、格式不正确或无效。

令牌撤销端点

在会话中持有刷新令牌的用户拥有类似于浏览器 Cookie 的东西。只要刷新令牌有效，他们就可以继续现有会话。在用户的 ID 或访问令牌到期后，您的应用程序可以使用刷新令牌来获取新的有效令牌，而

不是提示他们登录。但是，您可以从外部判断应该结束用户的会话，或者用户可能选择忘记其当前会话。此时，您可以撤销该刷新令牌，这样用户就无法再维持他们的会话。

`/oauth2/revoke` 端点撤销 Amazon Cognito 最初使用您提供的刷新令牌颁发的用户访问令牌。此端点还会撤销刷新令牌本身以及来自同一刷新令牌的所有后续访问和身份令牌。终端节点撤销令牌后，您无法使用已撤销的访问令牌来访问 Amazon Cognito 令牌进行 APIs 身份验证的访问权限。

POST /oauth2/revoke

`/oauth2/revoke` 端点只支持 `HTTPS POST`。用户池客户端直接对此端点发出请求，而不通过系统浏览器。

标头中的请求参数

`Authorization`

如果应用程序客户端具有客户端密钥，则客户端必须通过基本 HTTP 授权在其授权标头中传递 `client_id` 和 `client_secret`。密钥是基本 `Base64Encode(client_id:client_secret)`。

`Content-Type`

必须始终为 `'application/x-www-form-urlencoded'`。

正文中的请求参数

`token`

（必需）客户端要撤销的刷新令牌。请求还撤销 Amazon Cognito 使用此刷新令牌颁发的所有访问令牌。

必需。

`client_id`

（可选）您要撤消的令牌的应用程序客户端 ID。

如果客户端是公有的且没有密钥，则为必需。

撤消请求示例

此撤销请求会撤销没有客户端密钥的应用程序客户端的刷新令牌。注意请求正文中的 `client_id` 参数。

```
POST /oauth2/revoke HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
token=2YotnFZFEjr1zCsicMWpAA&
client_id=1example23456789
```

此撤销请求会撤销有客户端密钥的应用程序客户端的刷新令牌。请注意，`Authorization` 标头包含已编码的客户端 ID 和客户端密钥，但请求正文中没有 `client_id`。

```
POST /oauth2/revoke HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Accept: application/json
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
token=2YotnFZFEjr1zCsicMWpAA
```

撤消错误响应

成功的响应包含空正文。错误响应是一个带有 `error` 字段和 `error_description` 字段（在某些情况下）的 JSON 对象。

端点错误

- 如果令牌在请求中不存在或者应用程序客户端禁用了该特征，则您会收到 HTTP 400 和错误 `invalid_request`。
- 如果 Amazon Cognito 在撤销请求中发送的令牌不是刷新令牌，则您会收到 HTTP 400 和错误 `unsupported_token_type`。
- 如果客户端凭证无效，则您会收到 HTTP 401 和错误 `invalid_client`。
- 如果令牌已撤销或客户端提交的令牌无效，您会收到 HTTP 200 OK。

IdP SAML 断言端点

`/saml2/idpresponse` 接收 SAML 断言。在 service-provider-initiated（SP 启动的）登录中，您的应用程序不会直接与该端点交互——您的 SAML 2.0 身份提供商 (IdP) 使用他们的 SAML 响应将您的用户重定向到此处。对于 SP 发起的登录，请将 IdP 配置为将 `saml2/idpresponse` 的路径作为断言使

用者服务（ACS）URL。有关会话启动的更多信息，请参阅 SAML 会话在 Amazon Cognito 用户池中启动。

在 IdP 发起的登录中，用户通过您的 SAML 2.0 提供者登录后，在您的应用程序中调用此端点的请求。您的用户在浏览器中使用您的 IdP 登录，然后您的应用程序收集 SAML 断言并将其提交到此端点。您必须通过 HTTPS 在 HTTP POST 请求的正文中提交 SAML 断言。POST 请求的正文必须是 SAMLResponse 参数和 Relaystate 参数。有关更多信息，请参阅 实施 IDP 发起的 SAML 登录。

该saml2/idpresponse端点可以接受长度不超过 100,000 个字符的 SAML 断言。

POST **/saml2/idpresponse**

要在 IdP 发起的登录中使用 /saml2/idpresponse 端点，请生成一个 POST 请求，其中包含为您的用户池提供用户会话信息的参数。

- 他们希望登录到的应用程序客户端。
- 他们希望最终到达的回调 URL。
- 他们想要在用户的访问令牌中请求的 OAuth 2.0 范围。
- 启动登录请求的 IdP。

IdP 发起的请求正文参数

SAMLResponse

来自与您用户池中有效应用程序客户端和 IdP 配置关联的 IdP 的 Base64 编码 SAML 声明。

RelayState

RelayState 参数包含您原本要传递给 oauth2/authorize 端点的请求参数。有关这些参数的详细信息，请参阅对端点授权。

response_type

OAuth 2.0 的拨款类型。

client_id

应用程序客户端 ID。

redirect_uri

在 Amazon Cognito 授权用户之后，身份验证服务器将浏览器重定向到的 URL。

identity_provider

要将用户重定向到其中的身份提供者的名称。

idp_identifier

要将用户重定向到其中的身份提供者的标识符。

范围

您希望用户向授权服务器请求的 OAuth 2.0 范围。

具有正向响应的示例请求

示例：POST 请求

以下请求适用于在应用程序客户端 `1example23456789` 中从 IdP `MySAMLIdP` 获取用户的授权代码授予。用户`https://www.example.com`使用其授权码重定向到，该授权码可以兑换包含范围为 OAuth 2.0 和`openidemailphone`、的访问令牌的令牌。

```
POST /saml2/idpresponse HTTP/1.1
User-Agent: USER_AGENT
Accept: */*
Host: example.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded

SAMLResponse=[Base64-encoded SAML assertion]&RelayState=identity_provider
%3DMySAMLIdP%26client_id%3D1example23456789%26redirect_uri%3Dhttps%3A%2F
%2Fwww.example.com%26response_type%3Dcode%26scope%3Demail%2Bopenid%2Bphone
```

示例：响应

以下是对上一个请求的响应。

```
HTTP/1.1 302 Found
Date: Wed, 06 Dec 2023 00:15:29 GMT
Content-Length: 0
x-amz-cognito-request-id: 8aba6eb5-fb54-4bc6-9368-c3878434f0fb
Location: https://www.example.com?code=[Authorization code]
```

OAuth 2.0 补助金

Amazon Cognito 用户池 OAuth 2.0 授权服务器发布令牌以响应三种类型的 OAuth 2.0 授权授权。您可以为用户池中的每个应用程序客户端设置支持的授权类型。您不能在同一应用程序客户端中启用客户端凭证授予作为隐式 或授权码 授予。隐式和授权码授予的请求从对端点授权开始，而客户端凭证授予的请求从令牌端点开始。

授予授权代码

为了响应您的成功身份验证请求，授权服务器会在回调 URL 的 `code` 参数中添加一个授权码。然后，您必须与令牌端点交换 ID、访问令牌和刷新令牌的代码。要请求授权码授予，请在请求中将 `response_type` 设置为 `code`。有关示例请求，请参阅示例：授权码授权。Amazon Cognito 在授权码授权中支持代码交换证明密钥 (PKCE)。

授权码授予是最安全的授权授予形式。它不会直接向您的用户显示令牌内容。相反，您的应用程序负责检索和安全存储用户的令牌。在 Amazon Cognito 中，授权码授予是从授权服务器获取所有三种令牌类型（ID、访问和刷新）的唯一方法。您也可以通过 Amazon Cognito 用户池 API 从身份验证中获取所有三种令牌类型，但是 API 不使用除 `aws.cognito.signin.user.admin` 以外的范围发布访问令牌。

隐式授予

为了响应成功的身份验证请求，授权服务器会在 `access_token` 参数中附加一个访问令牌，并在您的回调 URL 的 `id_token` 参数中附加一个 ID 令牌。隐式授予不要求与令牌端点进行额外的交互。要请求隐式授予，请在请求中将 `response_type` 设置为 `token`。隐式授予仅生成 ID 和访问令牌。有关示例请求，请参阅示例：没有openid作用域的代币（隐式）授权。

隐式授予是旧版授权授予。与授权码授予不同，用户可以拦截和检查您的令牌。要防止通过隐式授予交付令牌，请将您的应用程序客户端配置为仅支持授权码授予。

客户端凭证

客户凭证是一种仅限授权的访问授权。 machine-to-machine要获得客户凭证授予，请绕过对端点授权，并直接向令牌端点生成请求。您的应用程序客户端必须具有客户端密钥，并且仅支持客户端凭证授予。为了响应您的成功请求，授权服务器会返回访问令牌。

来自客户端凭证授予的访问令牌是一种包含 OAuth 2.0 范围的授权机制。通常，令牌包含自定义作用域声明，这些声明授权 HTTP 操作受访问保护 APIs。有关更多信息，请参阅 作用域、M2M 以及 APIs 带资源服务器的场景。

客户凭证授予会增加您的 Amazon 账单费用。有关更多信息，请参阅 Amazon Cognito 定价。

刷新令牌

您可以直接从申请刷新令牌授权令牌端点。此授权将返回新的 ID 和访问令牌，以换取有效的刷新令牌。

有关这些赠款及其实施的更多视角，请参阅如何在 Amazon Cognito 中使用 OAuth 2.0：在Amazon 安全博客中了解不同的 OAuth 2.0 授权。

在授权代码授予中使用 PKCE

Amazon Cognito 在授权代码授予中支持代码交换的证明密钥（PKCE）身份验证。PKCE 是公共客户端的 OAuth 2.0 授权代码授予的扩展。PKCE 可防止兑换拦截的授权代码。

Amazon Cognito 如何使用 PKCE

要开始使用 PKCE 进行身份验证，您的应用程序必须生成一个唯一的字符串值。此字符串是代码验证程序，Amazon Cognito 使用该密钥值将请求初始授权授予的客户端与使用授权代码交换令牌的客户端进行比较。

您的应用程序必须对代码验证器字符串应用 SHA256 哈希值，并将结果编码为 base64。将经过哈希处理的字符串作为请求正文中的 `code_challenge` 参数传递到 对端点授权。当您的应用程序将授权代码交换为令牌时，它必须以纯文本形式包括代码验证程序字符串，作为请求正文中的 `code_verifier` 参数传递到 令牌端点。Amazon Cognito 对代码验证器执行相同的 hash-and-encode操作。Amazon Cognito 仅在确定代码验证程序产生的代码质询与在授权请求中收到的代码质询相同时，才会返回 ID 令牌、访问令牌和刷新令牌。

使用 PKCE 实现授权授予流程

1. 打开 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。如果您创建了用户池，则在向导期间将提示您设置应用程序客户端并配置托管登录。
   a. 如果您创建了新的用户池，请在引导式设置期间设置应用程序客户端并配置托管登录。
   b. 如果您配置现有用户池，请添加域和公共应用程序客户端（如果尚未添加）。
4. 生成一个随机的字母数字字符串 [通常是一个全局唯一标识符（UUID）]，以便为 PKCE 创建代码质询。此字符串是您将在发送给 令牌端点 的请求中提交的 `code_verifier` 参数的值。

5. 用 SHA256 算法对code_verifier字符串进行哈希处理。将哈希操作的结果编码为 base64。此字符串是您将在发送给 对端点授权 的请求中提交的 code_challenge 参数的值。

   以下 Python 示例生成 code_verifier 并计算 code_challenge：

```
#!/usr/bin/env python3

import random
from base64 import urlsafe_b64encode
from hashlib import sha256
from string import ascii_letters
from string import digits

# use a cryptographically strong random number generator source
rand = random.SystemRandom()

code_verifier = ''.join(rand.choices(ascii_letters + digits, k=128))
code_verifier_hash = sha256(code_verifier.encode()).digest()
code_challenge = urlsafe_b64encode(code_verifier_hash).decode().rstrip('=')

print(f"code challenge: {code_challenge}")
print(f"code verifier: {code_verifier}")
```

   以下是 Python 脚本的示例输出：

```
code challenge: Eh0mg-OZv7BAyo-tdv_vYamx1boOYDulDklyXoMDtLg
code verifier: 9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

6. 使用 PKCE 的授权码授权请求完成托管登录登录。以下是示例 URL：

```
https://mydomain.auth.us-east-1.amazoncognito.com/oauth2/authorize?
response_type=code&client_id=1example23456789&redirect_uri=https://
www.example.com&code_challenge=Eh0mg-OZv7BAyo-
tdv_vYamx1boOYDulDklyXoMDtLg&code_challenge_method=S256
```

7. 收集授权 code 并使用令牌端点将其兑换为令牌。以下是一个示例请求：

```
POST /oauth2/token HTTP/1.1
Host: mydomain.auth.us-east-1.amazoncognito.com
Content-Type: application/x-www-form-urlencoded
Content-Length: 296
```

```
redirect_uri=https%3A%2F%2Fwww.example.com&
client_id=1example23456789&
code=7378f445-c87f-400c-855e-0297d072ff03&
grant_type=authorization_code&
code_verifier=9D-aW_iygXrgQcWJd0y0tNVMPSXSChIc2xceDhvYVdGLCBk-
JWFTmBNjvKSdOrjTTYazOFbUmrFERrjWx6oKtK2b6z_x4_gHBDlr4K1mRFGyE8yA-05-_v7Dxf3EIYJH
```

8. 查看响应。响应将包含 ID 令牌、访问令牌和刷新令牌。有关使用 Amazon Cognito 用户池令牌的更多信息，请参阅了解用户池 JSON 网络令牌 (JWTs)。

托管登录和联盟错误响应

托管登录或联合登录中的登录过程可能会返回错误。以下是一些可能导致身份验证以错误结束的情况。

- 用户执行了您的用户池无法完成的操作。
- Lambda 触发器不以预期的语法响应。
- 您的身份提供者（IdP）返回错误。
- Amazon Cognito 无法验证您的用户提供的属性信息。
- 您的 IdP 未发送与所需属性对应的声明。

当 Amazon Cognito 遇到错误时，它通过以下方式之一进行通信。

1. Amazon Cognito 发送的重定向 URL 的请求参数中存在错误。
2. Amazon Cognito 在托管登录中显示错误。

Amazon Cognito 附加到请求参数的错误具有以下格式。

```
https://<Callback URL>/?error_description=error+description&error=error+name
```

如果您帮助用户在无法执行操作时提交错误信息，则要求他们捕获 URL 和文本或页面的屏幕截图。

> **Note**
>
> Amazon Cognito 错误描述不是固定字符串，您不应使用依赖于固定模式或格式的逻辑。

OIDC 和社交身份提供者错误消息

您的身份提供者可能会返回错误。当 OIDC 或 2.0 I OAuth dP 返回符合标准的错误时，Amazon Cognito 会将您的用户重定向到回传 URL，并将提供商错误响应添加到错误请求参数中。Amazon Cognito 将提供者名称和 HTTP 错误代码添加到现有的错误字符串中。

以下 URL 是从返回错误的 IdP 重定向到 Amazon Cognito 的示例。

```
https://www.amazon.com/?error_description=LoginWithAmazon+Error+-+400+invalid_request
+The+request+is+missing+a+required+parameter+%3A+client_secret&error=invalid_request
```

由于 Amazon Cognito 仅返回它从提供者处收到的内容，因此您的用户可能会看到这些信息的一部分。

如果用户在通过您的 IdP 进行初始登录时遇到问题，IdP 会直接向用户发送任何错误消息。当 Amazon Cognito 向您的 IdP 生成验证用户会话的请求时，它会向您的用户中继一条错误消息。Amazon Cognito 中继了 OAuth 来自以下终端节点的 OIDC IdP 错误消息。

`/token`

Amazon Cognito 交换 IdP 授权代码以获得访问令牌。

`/.well-known/openid-configuration`

Amazon Cognito 发现了通往发行者端点的路径。

`/.well-known/jwks.json`

为了验证用户的 JSON 网络令牌 (JWTs)，Amazon Cognito 会发现你的 IdP 用来签署令牌的 JSON 网络密钥 (JWKs)。

由于 Amazon Cognito 不会启动到可能返回 HTTP 错误的 SAML 2.0 提供者的出站会话，因此，用户在与 SAML 2.0 IdP 会话期间出现的错误不包括这种形式的提供者错误消息。

Amazon Cognito 身份池

Amazon Cognito 身份池是联合身份的目录，您可以用它交换 Amazon 凭证。身份池会为您的应用程序的用户生成临时 Amazon 证书，无论他们已登录还是您尚未识别他们的身份。通过 Amazon Identity and Access Management (IAM) 角色和策略，您可以选择要向用户授予的权限级别。用户能够以访客身份开始，然后检索您保留在 Amazon Web Services 服务中的资产。然后，他们可以通过第三方身份提供者登录，以解锁对您提供给注册会员的资产的访问权限。第三方身份提供商可以是 Apple 或 Google 等消费者（社交） OAuth 2.0 提供商、自定义 SAML 或 OIDC 身份提供商，也可以是您自己设计的自定义身份验证方案（也称为开发者提供商）。

Amazon Cognito 身份池的功能

签署请求 Amazon Web Services 服务

将 API 请求签名给亚马逊简单存储服务 (Amazon S3) Servic Amazon Web Services 服务 e 和亚马逊 DynamoDB。使用亚马逊 Pinpoint 和亚马逊等服务分析用户活动。 CloudWatch

使用基于资源的策略筛选请求

对于用户对资源的访问权限进行精细控制。将用户声明转换为 IAM 会话标签，并构建 IAM policy 以向用户的不同子集授予资源访问权限。

分配访客访问权限

对于尚未登录的用户，请将您的身份池配置为生成具有较窄访问权限范围的 Amazon 凭证。通过单点登录提供者对用户进行身份验证以提升其访问权限。

根据用户特征分配 IAM 角色

为所有经身份验证的用户分配一个 IAM 角色，或者根据每个用户的声明选择角色。

接受各种身份提供者

将 ID 或访问令牌、用户池令牌、SAML 断言或社交提供者 OAuth 令牌交换凭证。 Amazon

验证您自己的身份

执行您自己的用户验证，并使用您的开发者 Amazon 证书为您的用户颁发证书。

您可能已经有一个 Amazon Cognito 用户群体，用于为您的应用程序提供身份验证和授权服务。您可以将用户群体设置为身份池的身份提供者（IdP）。当你这样做时，你的用户可以通过你的用户池进行身

份验证 IdPs，将他们的声明整合到一个普通的 OIDC 身份令牌中，然后用该令牌兑换证书。 Amazon 然后，您的用户可以在签名的请求中向 Amazon Web Services 服务提供其凭证。

还可以将来自任何身份提供者的经身份验证的声明直接提供给身份池。Amazon Cognito 将 SAML 和 OIDC 提供商的用户声明自定义为短期证书的 API AssumeRoleWithWebIdentity请求。 OAuth

Amazon Cognito 用户群体类似于支持 SSO 的应用程序的 OIDC 身份提供者。对于具有最适合 IAM 授权的资源依赖关系的任何应用程序，身份池充当其 Amazon 身份提供者。

Amazon Cognito 身份池支持以下身份提供商：

- 公有提供者：设置 Login with Amazon 作为身份池 IdP、将 Facebook 设置为身份池 IdP、将 Google 设置为身份池 IdP、使用 Apple 作为身份池 IdP 来设置登录、Twitter。
- Amazon Cognito 用户池
- 设置 OIDC 提供者作为身份池 IdP
- 设置 SAML 提供者作为身份池 IdP
- 经开发人员验证的身份

有关 Amazon Cognito 身份池区域可用性的信息，请参阅Amazon 服务区域可用性。

有关 Amazon Cognito 身份池的更多信息，请参阅以下主题。

主题

- 身份池控制台概述
- 身份池身份验证流程
- IAM 角色
- Amazon Cognito 身份池的安全最佳实践
- 将属性用于访问控制
- 使用基于角色的访问控制
- 获取凭证
- 使用临时 Amazon Web Services 服务 凭证进行访问
- 身份池第三方身份提供者
- 经开发人员验证的身份
- 将未经身份验证的用户切换为经过身份验证的用户

身份池控制台概述

Amazon Cognito 身份池为访客用户（未经身份验证）和已通过身份验证并收到令牌的用户提供临时 Amazon 证书。身份池是指与您的外部身份提供者关联的用户标识符的存储。

要了解身份池的特征和选项，一种方法是在 Amazon Cognito 控制台中创建一个身份池。您可以探索不同设置对身份验证流程、基于角色和基于属性的访问控制以及访客访问的影响。接下来，您可以继续阅读本指南的后面章节，并向您的应用程序添加相应的组件，以便可以实施身份池身份验证。

主题

- 创建 身份池
- 用户 IAM 角色
- 经过身份验证和未经身份验证的身份
- 激活或停用访客访问权限
- 更改与身份类型关联的角色
- 编辑身份提供者
- 删除身份池
- 从身份池删除身份
- 将 Amazon Cognito Sync 与身份池一起使用

创建 身份池

在控制台中创建新的身份池

1. 登录 Amazon Cognito 控制台并选择身份池。
2. 选择创建身份池。
3. 在配置身份池信任中，选择将您的身份池设置为经过身份验证的访问权限和/或访客访问权限。
   - 如果您选择了经过身份验证的访问权限，请在身份池中选择一个或多个您要设置为经过身份验证的身份来源的身份类型。如果您配置了自定义开发人员提供者，则在创建身份池后无法对其进行修改或删除。
4. 在配置权限中，为身份池中经过身份验证的用户或访客用户选择原定设置 IAM 角色。
   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如

myidentitypool_authenticatedrole。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

b. 如果您的 Amazon Web Services 账户 角色中已有要使用的角色，则可以选择使用现有 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 cognito-identity.amazonaws.com。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅 角色信任和权限。

5. 在 Connect 身份提供商中，输入您在配置身份池信任中选择的身份提供商 (IdPs) 的详细信息。系统可能会要求您提供 OAuth 应用程序客户端信息、选择 Amazon Cognito 用户池、选择 IAM IdP 或输入开发者提供商的自定义标识符。

   a. 为每个 IdP 选择角色设置。您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 cognito:preferred_role 声明的更多信息，请参阅将优先级值分配到组。

      i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

      ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

   b. 您可以为每个 IdP 配置访问控制属性。访问控制属性将用户声明映射到 Amazon Cognito 应用于其临时会话的主体标签。您可以构建 IAM policy，以根据应用于用户会话的标签来筛选用户访问权限。

      i. 如果不应用主体标签，请选择非活动。

      ii. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

      iii. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

6. 在配置属性中，在身份池名称下输入名称。

7. 在基本（经典）身份验证下，选择是否要激活基本流程。启用基本流程后，您可以绕过为自己选择的角色 IdPs ，AssumeRoleWithWebIdentity直接致电。有关更多信息，请参阅 身份池身份验证流程。

8. 如果要将标签应用到身份池，请在标签下选择添加标签。

9. 在查看并创建中，确认您为新身份池所做的选择。选择编辑以返回向导并更改任何设置。完成后，选择创建身份池。

用户 IAM 角色

IAM 角色定义用户访问 Amazon 资源的权限，例如Amazon Cognito Sync。您的应用程序用户将担任您创建的角色。您可以为经过身份验证和未经身份验证的用户指定不同角色。要了解有关 IAM 角色的更多信息，请参阅 IAM 角色。

经过身份验证和未经身份验证的身份

Amazon Cognito 身份池同时支持经过身份验证和未经身份验证的身份。经过身份验证的身份属于已通过任何受支持的身份提供商进行身份验证的用户。未经身份验证的身份通常属于来宾用户。

- 要使用公共登录提供商配置经过身份验证的身份，请参阅 身份池第三方身份提供者。
- 要配置您自己的后端身份验证流程，请参阅 经开发人员验证的身份。

激活或停用访客访问权限

Amazon Cognito 身份池访客访问（未经身份验证的身份）为未向身份提供者进行身份验证的用户提供唯一标识符和 Amazon 证书。如果应用程序允许未登录的用户进行访问，则您可以针对未经身份验证的身份激活访问权限。要了解更多信息，请参阅Amazon Cognito 身份池入门。

更新身份池中的访客访问权限

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到访客访问权限。在目前不支持访客访问的身份池中，状态为非活动。
   a. 如果访客访问权限处于活动状态并且您想将其停用，请选择停用。
   b. 如果访客访问权限处于非活动状态而您想要将其激活，请选择编辑。
      - 为身份池中的访客用户选择原定设置 IAM 角色。
        A. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

B. 如果您的 Amazon Web Services 账户 角色中已有要使用的角色，则可以选择使用现有 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅 角色信任和权限。

C. 选择保存更改。

D. 要激活访客访问权限，请在用户访问权限选项卡中选择激活。

更改与身份类型关联的角色

身份池中的每个身份要么经过身份验证，要么未经过身份验证。经过身份验证的身份属于通过公共登录提供商（Amazon Cognito 用户池、Login with Amazon、Sign in with Apple、Facebook、Google、SAML 或任何 OpenID Connect 提供商）或开发人员提供商（自己的后端身份验证流程）验证身份的用户。未经身份验证的身份通常属于来宾用户。

每个身份类型都有一个分配的角色。此角色附有策略，规定 Amazon Web Services 服务 该角色可以访问哪个角色。Amazon Cognito 接收请求后，服务将确定身份类型、确定分配给该身份类型的角色，并使用附加到该角色的策略进行响应。通过修改策略或为身份类型分配不同的角色，您可以控制哪些 Amazon Web Services 服务 身份类型可以访问。要查看或修改与身份池中与角色关联的策略，请参阅 Amazon IAM 控制台。

更改身份池的原定设置经过身份验证或未经身份验证的角色

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到访客访问权限或经过身份验证的访问权限。在当前未为该访问类型配置的身份池中，状态为非活动。选择编辑。
4. 为身份池中的访客用户或经过身份验证的用户选择原定设置 IAM 角色。

   a. 如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

   b. 如果您的 Amazon Web Services 账户 角色中已有要使用的角色，则可以选择使用现有 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供

证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅 角色信任和权限。

5. 选择保存更改。

编辑身份提供者

如果您允许用户通过使用者身份提供者（例如，Amazon Cognito 用户群体、Login with Amazon、通过 Apple 登录、Facebook 或 Google）进行身份验证，则您可以在 Amazon Cognito 身份池（联合身份）控制台中指定应用程序标识符。上述操作会将应用程序 ID (由公共登录提供商提供) 与身份池关联。

您还可以从此页面为每个提供商配置身份验证规则。每个提供商最多可以有 25 个规则。规则按您为各个提供商保存的顺序应用。有关更多信息，请参阅 使用基于角色的访问控制。

> **Warning**
>
> 更改身份池中关联的 IdP 应用程序 ID 可防止现有用户通过该身份池进行身份验证。有关更多信息，请参阅 身份池第三方身份提供者。

更新身份池身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。如果要添加新的 IdP，请选择添加身份提供者。
   - 如果您选择添加身份提供者，请选择要添加的身份类型之一。
4. 要更改应用程序 ID，请在身份提供者信息中选择编辑。
5. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请在角色设置中选择编辑。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。使用 Amazon Cognito 用户群体 IdP，还可以选择令牌中包含 preferred_role 声明的角色。有关 `cognito:preferred_role` 声明的更多信息，请参阅将优先级值分配到组。

   i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。

   ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

6. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请在访问控制属性中选择编辑。

   a. 如果不应用主体标签，请选择非活动。

   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

7. 选择保存更改。

删除身份池

您不能撤消身份池删除。删除身份池后，所有依赖该身份池的应用程序和用户将停止工作。

删除身份池

1. 从 Amazon Cognito 控制台中选择身份池。选中要删除的身份池旁边的单选按钮。
2. 选择删除。
3. 输入或粘贴身份池的名称，然后选择删除。

> **Warning**
>
> 选择删除按钮后，您将永久删除身份池和其中包含的所有用户数据。删除身份池将导致使用身份池的应用程序和其他服务停止工作。

从身份池删除身份

当您从身份池中删除身份时，您会删除 Amazon Cognito 为该联合用户存储的身份信息。当用户再次请求凭证时，如果身份池仍然信任用户的身份提供者，则用户会收到新的身份 ID。您无法撤消此操作。

删除身份

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选中要删除的身份旁边的复选框，然后选择删除。确认您要删除这些身份，然后选择删除。

将 Amazon Cognito Sync 与身份池一起使用

Amazon Cognito Sync 是一个 Amazon Web Services 服务 和客户端库，它使跨设备同步与应用程序相关的用户数据成为可能。Amazon Cognito 可以跨移动设备和 Web 同步用户配置文件数据，无需使用您自己的后端。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据。如果您设置推送同步，您可在更新可用时立即通知其他设备。

管理数据集

如果您在应用程序中实施了 Amazon Cognito Sync 功能，则 Amazon Cognito 身份池控制台允许您手动创建和删除各个身份的数据集和记录。对于您在 Amazon Cognito 身份池控制台中对身份的数据集或记录做出的任何更改，只有当您在控制台中选择 Synchronize（同步）后才会保存。直到身份调用 Synchronize（同步）后，终端用户才能看到更改。一旦刷新特定身份的列表数据集页面，从其它设备同步的有关各个身份的数据即会显示。

为身份创建数据集

Amazon Cognito Sync 将数据集与一个身份关联起来。您可以在数据集中填充该身份所代表的用户的身份信息，然后将该信息同步到用户的所有设备。

将数据集和数据集记录添加到身份

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选择要编辑的身份。
4. 在数据集中，选择创建数据集。
5. 输入数据集名称并选择创建数据集。
6. 如果您想向数据集添加记录，请从身份详细信息中选择您的数据集。在记录中，选择创建记录。
7. 输入记录的键和值。选择确认。重复此操作以添加更多记录。

删除与身份关联的数据集

从身份中删除数据集及其记录

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份浏览器选项卡。
3. 选择包含要删除的数据集的身份。
4. 在数据集中，选择要删除的数据集旁边的单选按钮。
5. 选择删除。查看您的选择，然后再次选择删除。

批量发布数据

批量发布可用于将存储在 Amazon Cognito Sync 存储中的数据导出到 Amazon Kinesis Stream。有关如何批量发布所有流的说明，请参阅 实施 Amazon Cognito Sync 流。

激活推送同步

Amazon Cognito 会自动跟踪身份和设备之间的关联。通过使用推送同步功能，可以确保在身份数据发生更改时通知给定身份的每个实例。推送同步可以确保，只要身份的数据集发生更改，与该身份关联的所有设备就会收到一个静音推送通知，通知它们所发生的更改。

您可以在 Amazon Cognito 控制台中激活推送同步。

激活推送同步

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择身份池属性选项卡。
3. 在推送同步中，选择编辑
4. 选择激活与身份池的推送同步。
5. 选择您在当前 Amazon Web Services 区域中创建的 Amazon Simple Notification Service（Amazon SNS）平台应用程序之一。Amazon Cognito 向您的平台应用程序发布推送通知。选择创建平台应用程序以导航到 Amazon SNS 控制台并创建一个新的应用程序。
6. 要发布到平台应用程序，Amazon Cognito 将代入您的 Amazon Web Services 账户中的 IAM 角色。如果您希望 Amazon Cognito 为您创建一个具有基本权限并与您的身份池建立信任关系的新角色，请选择创建新的 IAM 角色。例如，输入 IAM 角色名称以标识您的新角色，例如 `myidentitypool_authenticatedrole`。选择查看策略文档以查看 Amazon Cognito 将分配给新 IAM 角色的权限。

7. 如果您的 Amazon Web Services 账户 角色中已有要使用的角色，则可以选择使用现有 IAM 角色。您必须将您的 IAM 角色信任策略配置为包括 `cognito-identity.amazonaws.com`。配置您的角色信任策略，以仅允许 Amazon Cognito 在提供证据证明请求来自您的特定身份池中经过身份验证的用户时，才代入该角色。有关更多信息，请参阅 角色信任和权限。
8. 选择保存更改。

设置 Amazon Cognito Streams

Amazon Cognito Streams 让开发人员能够控制和了解他们存储在 Amazon Cognito Sync 中的数据。开发人员现在可以配置 Kinesis 流以接收数据形式的事件。Amazon Cognito 可以实时向您拥有的 Kinesis 流推送每个数据集更改。有关如何在 Amazon Cognito 控制台中设置 Amazon Cognito Streams 的说明，请参阅实施 Amazon Cognito Sync 流。

设置 Amazon Cognito Events

Amazon Cognito Events 允许您运行 Amazon Lambda 函数以响应 Amazon Cognito Sync 中的重要事件。当数据集得到同步时，Amazon Cognito Sync 会引发同步触发事件。当用户更新数据时，您可以使用同步触发事件采取行动。有关从控制台设置 Amazon Cognito Events 的说明，请参阅使用 Amazon Cognito Events 自定义工作流。

要了解更多信息 Amazon Lambda，请参阅Amazon Lambda。

身份池身份验证流程

Amazon Cognito 可帮助您为终端用户创建在多个设备和平台间保持一致的唯一标识符。Amazon Cognito 还会向您的应用程序提供临时的、权限有限的凭证以访问资源。 Amazon 此页面介绍有关 Amazon Cognito 中的身份验证如何工作的基础知识，并解释了身份池中身份的生命周期。

外部提供商身份验证流程

使用 Amazon Cognito 进行身份验证的用户将通过一个多步骤流程来引导启动其凭证。Amazon Cognito 提供两个不同的通过公有提供商进行身份验证的流程：增强型流程和基本流程。

完成其中一个流程后，您可以访问由您的角色访问策略定义的其他 Amazon Web Services 服务 流程。默认情况下，Amazon Cognito 控制台创建可访问 Amazon Cognito Sync 存储和 Amazon Mobile Analytics 的角色。有关如何授予额外访问权限的更多信息，请参阅 IAM 角色。

身份池接受来自提供者的以下构件：

提供商	身份验证构件
Amazon Cognito 用户池	ID 令牌
OpenID Connect（OIDC）	ID 令牌
SAML 2.0	SAML 断言
社交提供者	访问令牌

增强型（简化的）身份验证流程

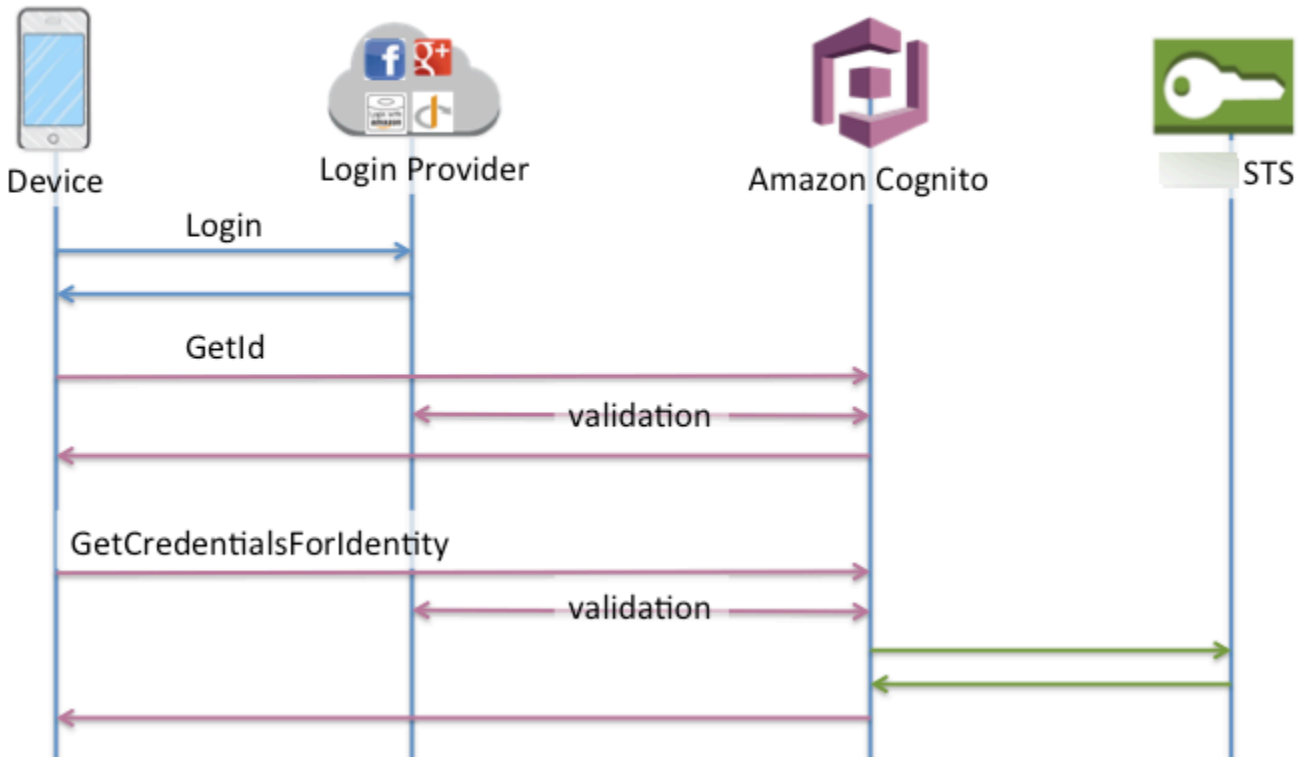
当您使用增强版身份验证流程时，您的应用程序会首先在请求中提供来自授权的 Amazon Cognito 用户池或第三方身份提供商的身份验证证明。GetId

1. 您的应用程序显示 GetId 请求中来自已授权 Amazon Cognito 用户池或第三方身份提供者的身份验证证明（JSON Web 令牌或 SAML 断言）。
2. 您的身份池返回身份 ID。
3. 您的应用程序在GetCredentialsForIdentity请求中将身份 ID 与相同的身份验证证明相结合。
4. 您的身份池会返回 Amazon 证书。
5. 您的应用程序使用临时证书签署 Amazon API 请求。

增强型身份验证管理您的身份池配置中的 IAM 角色选择和凭证检索的逻辑。您可以配置身份池以选择默认角色，将基于属性的访问权限控制（ABAC）或基于角色的访问权限控制（RBAC）原则应用于角色选择。来自增强身份验证的 Amazon 凭证有效期为一小时。

增强型身份验证中的操作顺序

1. `GetId`
2. `GetCredentialsForIdentity`



基本（经典）工作流程

当您使用基本身份验证流程时，

1. 您的应用程序显示 GetId 请求中来自已授权 Amazon Cognito 用户池或第三方身份提供者的身份验证证明（JSON Web 令牌或 SAML 断言）。
2. 您的身份池返回身份 ID。
3. 您的应用程序在GetOpenIdToken请求中将身份 ID 与相同的身份验证证明相结合。
4. `GetOpenIdToken`返回由您的身份池颁发的新 OAuth 2.0 令牌。
5. 您的应用程序在AssumeRoleWithWebIdentity请求中显示新令牌。
6. Amazon Security Token Service (Amazon STS) 返回 Amazon 凭证。
7. 您的应用程序使用临时证书签署 Amazon API 请求。

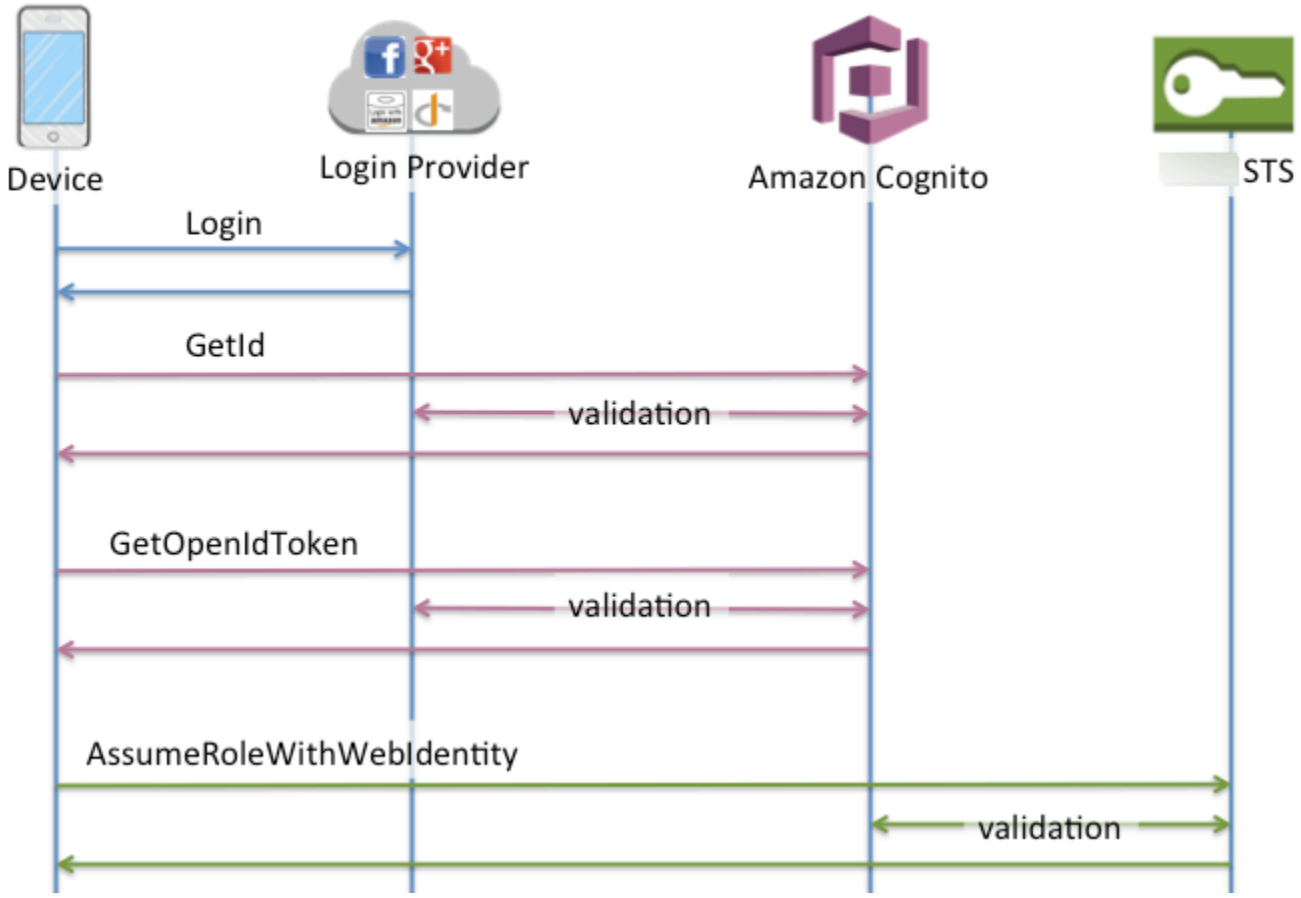
通过基本型工作流，您可以更精细地控制分发给用户的凭证。增强型身份验证流程的 `GetCredentialsForIdentity` 请求根据访问令牌的内容请求角色。经典工作流程中的`AssumeRoleWithWebIdentity`请求使您的应用程序能够更好地请求您配置了足够信任策略的任何 Amazon Identity and Access Management 角色的凭证。您也可以请求自定义角色会话持续时间。

您可以在没有角色映射的用户池中使用基本身份验证流程进行登录。这种类型的身份池没有默认的已验证角色或未经身份验证的角色，也没有配置基于角色或基于属性的访问控制。当您在使用角色映射的身份池中尝试 `GetOpenIdToken` 时，会收到以下错误。

```
Basic (classic) flow is not supported with RoleMappings, please use enhanced flow.
```

基本身份验证中的操作顺序

1. `GetId`
2. `GetOpenIdToken`
3. `AssumeRoleWithWebIdentity`



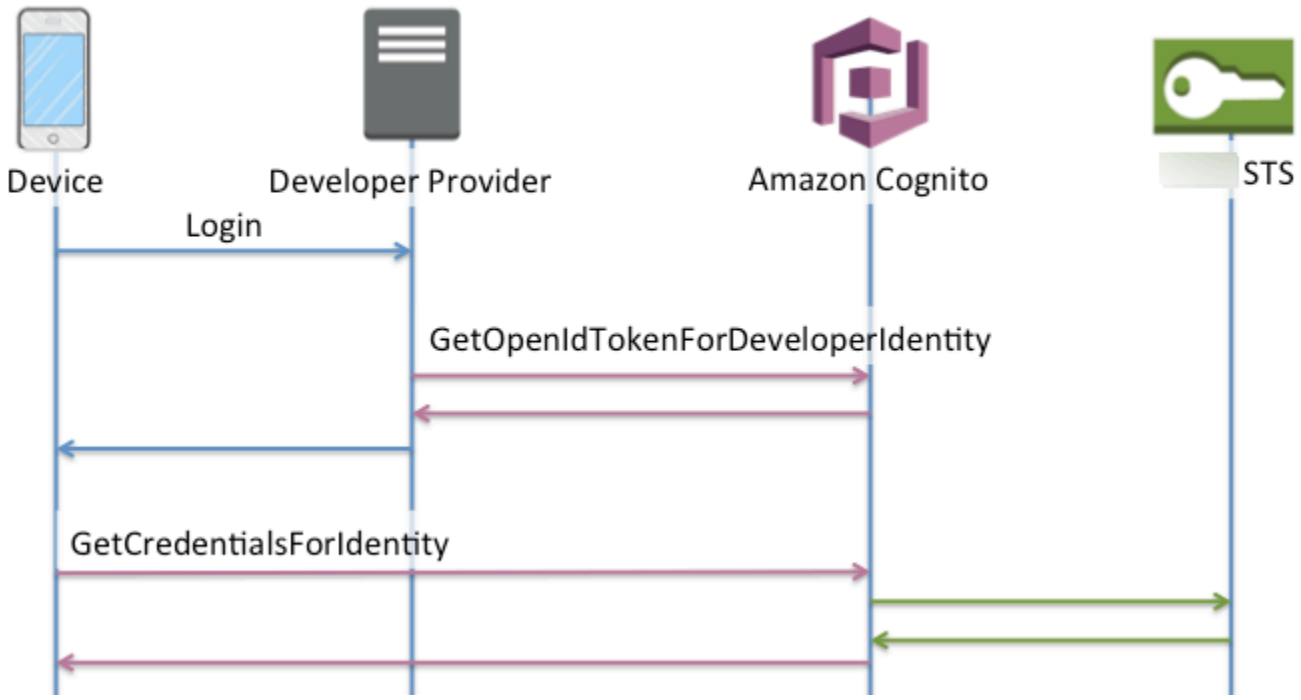
已经过开发人员验证的身份验证流程

使用经开发人员验证的身份时，客户端将使用包括 Amazon Cognito 外部代码的不同身份验证流程，在您自己的身份验证系统中验证用户。Amazon Cognito 外部代码如此处所示。

增强型身份验证流程

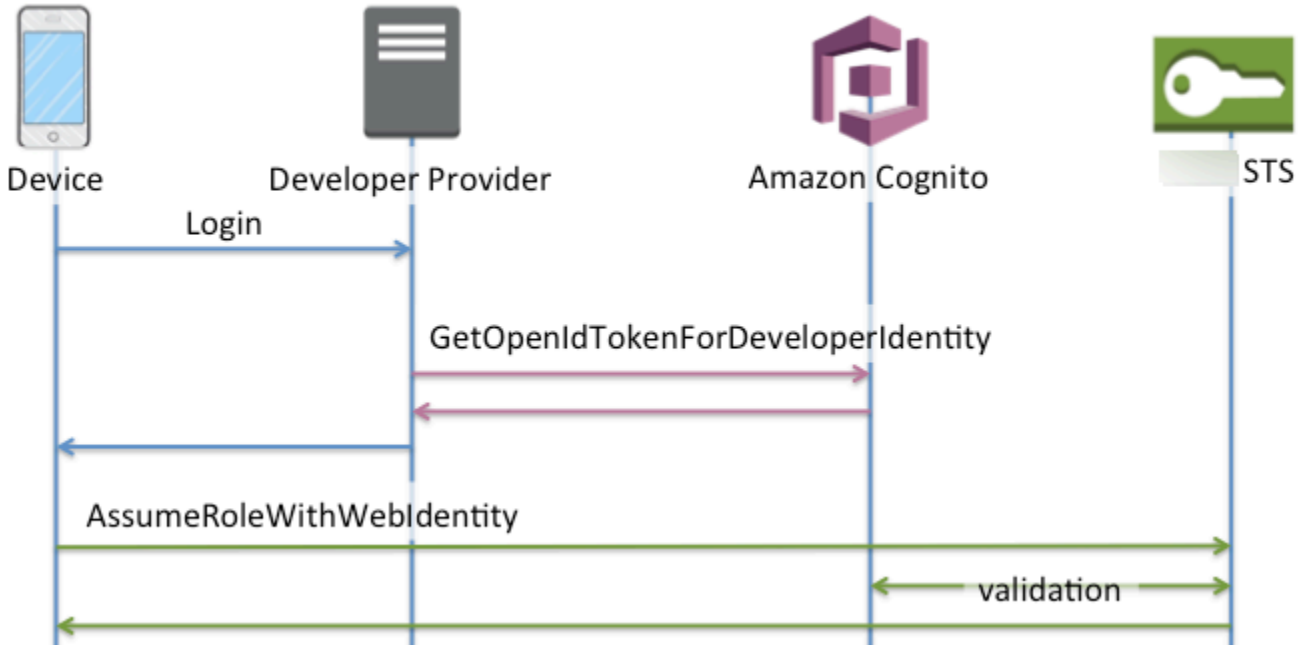
开发人员提供者的增强型身份验证中的操作顺序

1. 通过开发人员提供商登录（Amazon Cognito 外部代码）
2. 验证用户登录（Amazon Cognito 外部代码）
3. GetOpenIdTokenForDeveloperIdentity
4. GetCredentialsForIdentity



开发人员提供者的基本身份验证中的操作顺序

1. 在身份池之外实施逻辑来登录，并生成开发人员提供者标识符。
2. 检索存储的服务器端 Amazon 凭证。
3. 在使用授权 Amazon 凭证签名GetOpenIdTokenForDeveloperIdentity的 API 请求中发送开发者提供商标识符。
4. 使用请求应用程序凭证AssumeRoleWithWebIdentity。



我应该使用哪种身份验证流程？

增强型流程是更安全的选择，开发人员的工作量更少：

- 增强型流程降低了 API 请求的复杂性、大小和频率。
- 您的应用程序无需向 Amazon STS发出其他 API 请求。
- 您的身份池会评估您的用户应收到的 IAM 角色凭证。您无需在客户端中嵌入用于角色选择的逻辑。

> **Important**
>
> 创建新的身份池时，最好不要默认激活基本（经典）身份验证。要实施基本身份验证，请先评估您的 IAM 角色与 Web 身份之间的信任关系。然后在您的客户端中构建角色选择逻辑，并保护客户端不被用户修改。

基本身份验证流程将 IAM 角色选择的逻辑委托给您的应用程序。在此流程中，Amazon Cognito 会验证您的用户经过身份验证或未经身份验证的会话，并颁发一个令牌，您可以用该令牌交换证书。 Amazon STS用户可以将基本身份验证中的令牌交换为信任您的身份池和 `amr` 或处于经过身份验证/未经身份验证状态的任何 IAM 角色。

同样，您要明白，开发人员身份验证是一种绕过身份提供者身份验证的快捷方式。Amazon Cognito 信任授权GetOpenIdTokenForDeveloperIdentity请求的 Amazon 凭证，无需对请求内容进行额外验证。保护用于授权开发人员身份验证的密钥，防止用户访问这些信息。

API 摘要

GetId

GetId API 调用是在 Amazon Cognito 中建立新身份所必需的第一个调用。

未经身份验证的访问

Amazon Cognito 能够在您的应用程序中授权未经身份验证的访客访问。如果您的身份池中已启用此功能，用户可以随时通过 `GetId` API 请求新的身份 ID。该应用程序应缓存此身份 ID，以对 Amazon Cognito 发出后续调用。 Amazon 移动设备 SDKs 和浏览器 JavaScript 中的 S Amazon DK 都有凭据提供商，可以为您处理缓存。

经过身份验证的访问

当你将应用程序配置为支持公共登录提供商（Facebook、Google+、使用亚马逊登录或用苹果登录）时，用户还可以提供令牌（或 OAuth OpenID Connect），用于在这些提供商中识别他们。在对 `GetId` 的调用中使用时，Amazon Cognito 创建一个经过身份验证的新身份，或者是返回已与该特定登录关联的身份。Amazon Cognito 通过向提供商验证令牌并确保以下各项来实现此目的：

- 令牌有效且来自已配置的提供商。
- 令牌未过期。
- 令牌与使用该提供商创建的应用程序标识符（如 Facebook 应用程序 ID）匹配。
- 令牌与用户标识符匹配。

GetCredentialsForIdentity

在建立身份 ID 后，即可调用 GetCredentialsForIdentity API。因此，此操作在功能上等同于调用GetOpenIdToken。AssumeRoleWithWebIdentity

要让 Amazon Cognito 代表您调用 `AssumeRoleWithWebIdentity`，您的身份池必须具有与之关联的 IAM 角色。为此，您可以使用 Amazon Cognito 控制台，也可以通过 SetIdentityPoolRoles 手动操作实现此目的。

GetOpenIdToken

在建立身份 ID 后，即可发出 GetOpenIdToken API 请求。 IDs 在您第一次请求后缓存身份，然后使用启动该身份的后续基本（经典）会话`GetOpenIdToken`。

对 `GetOpenIdToken` API 请求的响应是 Amazon Cognito 生成的令牌。您可以将此令牌作为 AssumeRoleWithWebIdentity 请求中的 `WebIdentityToken` 参数提交。

在提交 OpenID 令牌之前，请在您的应用程序中进行验证。你可以在 SDK 中使用 OIDC 库或者类似的库 aws-jwt-verify以确认 Amazon Cognito 已发行该代币。OpenID 令牌的签名密钥 ID 或`kid`，是亚马逊 Cognito Identity 中列出的密钥 ID 之一 jwks_uri 文档 †。这些密钥可能会发生变化。验证 Amazon Cognito 身份令牌的函数应定期更新它在 jwks_uri 文档中的密钥列表。Amazon Cognito 在 jwks_uri 缓存控制响应标头中设置刷新时长，而 `max-age` 当前设置为 30 天。

未经身份验证的访问

要为未经身份验证的身份获取令牌，您只需身份 ID 本身。无法为经身份验证的身份或已停用的身份获取未经身份验证的令牌。

经过身份验证的访问

如果您有一个经过身份验证的身份，您必须为已与该身份关联的登录名传递至少一个令牌。在 `GetOpenIdToken` 调用期间，所有传入的令牌都必须通过之前提到的同一验证；如果有任何令牌失败，整个调用都会失败。`GetOpenIdToken` 调用的响应中还包括身份 ID。这是因为您传入的身份 ID 可能不是返回的那个身份 ID。

关联登录名

如果您为尚未与任何身份关联的登录名提交令牌，该登录名将视为已“关联”到关联身份。您只能为每个公共提供商链接一个登录名。如果尝试将多个登录名链接到一个公共提供商，将导致 `ResourceConflictException` 错误响应。如果登录名只链接到一个现有身份，则 `GetOpenIdToken` 返回的身份 ID 将与传入的相同。

合并身份

如果您为目前未链接到给定身份，但链接到另一身份的登录名传入令牌，这两个身份将合并。合并后，将返回parent/owner of all associated logins and the other is disabled. In this case, the identity ID of the parent/owner一个身份。如果此值不同，则必须更新本地缓存。 Amazon 移动版 SDKs 或浏览器版 Amazon SDK JavaScript 中的提供商会为您执行此操作。

GetOpenIdTokenForDeveloperIdentity

当使用经过开发者身份验证的身份时，该GetOpenIdTokenForDeveloperIdentity操作取代了GetOpenIdToken从设备中使用GetId和从设备中使用。由于您的应用程序使用 Amazon 凭证对此 API 操作的请求进行签名，因此 Amazon Cognito 相信请求中提供的用户标识符是有效的。开发人员身份验证会替代 Amazon Cognito 对外部提供者执行的令牌验证。

此 API 的有效载荷包括 `logins` 映射。此映射必须包含您的开发人员提供者密钥，以及作为系统中用户标识符的值。如果用户标识符尚未关联到现有身份，Amazon Cognito 会创建新的身份，并为

该身份返回新身份 ID 以及 OpenID Connect 令牌。如果用户标识符已关联，Amazon Cognito 返回预先存在的身份 ID 和 OpenID Connect 令牌。 IDs 在您第一次请求后缓存开发者身份，然后使用启动该身份的后续基本（经典）会话`GetOpenIdTokenForDeveloperIdentity`。

对 `GetOpenIdTokenForDeveloperIdentity` API 请求的响应是 Amazon Cognito 生成的令牌。您可以将此令牌作为 `AssumeRoleWithWebIdentity` 请求中的 `WebIdentityToken` 参数提交。

在提交 OpenID Connect 令牌之前，请在您的应用程序中进行验证。你可以在 SDK 中使用 OIDC 库或者类似的库 aws-jwt-verify以确认 Amazon Cognito 已发行该代币。OpenID Connect 令牌的签名密钥 ID 或 `kid` 是 Amazon Cognito 身份 jwks_uri 文档中列出的密钥之一。这些密钥可能会发生变化。验证 Amazon Cognito 身份令牌的函数应定期更新它在 jwks_uri 文档中的密钥列表。Amazon Cognito 在 jwks_uri `cache-control` 响应标头中设置刷新时长，`max-age` 当前设置为 30 天。

关联登录名

与外部提供商一样，提供尚未与身份关联的额外登录名会将这些登录名隐式关联到该身份。如果您将一个外部提供者登录名链接到一个身份，用户可以对该提供者使用外部提供者身份验证流程。但是，他们无法在调用 `GetId` 或 `GetOpenIdToken` 时使用登录映射中的开发人员提供者名称。

合并身份

借助已经过开发人员验证的身份，Amazon Cognito 支持隐式合并以及通过 MergeDeveloperIdentities API 进行显式合并。通过显式合并，您可以使用系统中的用户标识符将两个身份标记为单个身份。如果您提供了源和目标用户标识符，Amazon Cognito 会将其合并。您下次为任一用户标识符请求 OpenID Connect 令牌时，系统都会返回同一身份 ID。

AssumeRoleWithWebIdentity

在你获得 OpenID Connect 令牌后，你可以通过 AssumeRoleWithWebIdentityAPI 请求将其换成临时 Amazon 证书 Amazon Security Token Service ()Amazon STS。

由于可以创建的身份数量没有限制，所以您务必要了解向用户授予的权限。为应用程序设置不同的 IAM 角色：一个用于未经身份验证的用户，一个用于经过身份验证的用户。当您首次设置身份池时，Amazon Cognito 控制台会创建默认角色。实际上没有向这些角色授予任何权限。修改这些角色以满足您的需求。

了解有关 角色信任和权限 的更多信息。

† 原定设置 Amazon Cognito 身份 jwks_uri 文档包含有关在大多数 Amazon Web Services 区域中用于签署身份池令牌的密钥的信息。以下区域有不同的 jwks_uri 文档。

Amazon Cognito Identity JSON web key URIs in other Amazon Web Services 区域

Amazon Web Services 区域	jwks_uri 文档的路径
Amazon GovCloud （美国西部）	`https://cognito-identity.us-gov-west-1.amazonaws.com/.well-known/jwks_uri`
中国（北京）	`https://cognito-identity.cn-north-1.amazonaws.com.cn/.well-known/jwks_uri`
选择加入区域，例如欧洲（米兰）和非洲（开普敦）	`https://cognito-identity.` *`Region`*`.amazonaws.com/.well-known/jwks_uri`

您还可以从颁发者推断出 jwks_uri，或者从 Amazon Cognito 推断出在 OpenID 令牌中收到的 `iss`。OIDC 标准发现端点 `<issuer>/.well-known/openid-configuration` 列出了您的令牌的 jwks_uri 的路径。

IAM 角色

在创建身份池时，系统会提示您更新用户代入的 IAM 角色。IAM 角色的工作方式如下：当用户登录应用程序时，Amazon Cognito 为用户生成临时 Amazon 凭证。这些临时凭证与特定 IAM 角色相关联。通过 IAM 角色，您可以定义一组权限，用来访问您的 Amazon 资源。

您可以为经过身份验证的用户和未经身份验证的用户指定默认 IAM 角色。此外，您可以定义规则，以便基于用户 ID 令牌中的声明为每个用户选择角色。有关更多信息，请参阅 使用基于角色的访问控制。

默认情况下，Amazon Cognito 控制台将创建可提供访问 Amazon Mobile Analytics 和 Amazon Cognito Sync 的权限的角色。或者，您也可以选择使用现有的 IAM 角色。

修改 IAM 角色以允许或限制对其他服务的访问。为此，请登录 IAM 控制台。然后，选择 Roles（角色），选择一个角色。Permissions（权限）选项卡中会列出选定角色所附加的策略。您可以选择相应的 Manage Policy（管理策略）链接自定义访问策略。要了解如何使用和定义策略的更多信息，请参阅 IAM 策略概述。

> **Note**
>
> 作为最佳实践，定义策略时应遵循授予最低权限的原则。换言之，策略只包含用户执行其任务所需的权限。有关更多信息，请参阅 IAM 用户指南中的授予最低权限。
> 请记住，未经验证的身份会被未登录应用的用户所利用。通常情况下，为未经验证的身份分配的权限应该比为经过验证的身份分配的权限更严格。

主题

- 设置信任策略
- 访问策略
- 角色信任和权限

设置信任策略

Amazon Cognito 使用 IAM 角色为应用程序的用户生成临时凭证。对权限的访问由角色的信任关系控制。了解有关 角色信任和权限 的更多信息。

呈现给 Amazon STS 的令牌由身份池生成，身份池将用户池、社交或 OIDC 提供商令牌或 SAML 断言转换为自己的令牌。身份池令牌包含一个 aud 声明，即身份池 ID。

以下示例角色信任策略允许联合服务主体`cognito-identity.amazonaws.com`调用 Amazon STS API `AssumeRoleWithWebIdentity`。仅当 API 请求中的身份池令牌具有以下声明时，请求才会成功。

1. 一个 aud 声明（身份池 ID `us-west-2:abcdefg-1234-5678-910a-0e8443553f95`）。
2. 一个 amr 声明（`authenticated`），当用户已登录并且不是访客用户时添加。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"
            },
            "Action": "sts:AssumeRoleWithWebIdentity",
```

```
            "Condition": {
                "StringEquals": {
                    "cognito-identity.amazonaws.com.cn:aud": "us-
west-2:abcdefg-1234-5678-910a-0e8443553f95"
                },
                "ForAnyValue:StringLike": {
                    "cognito-identity.amazonaws.com.cn:amr": "authenticated"
                }
            }
        }
    ]
}
```

基本（经典）身份验证中 IAM 角色的信任策略

您必须至少应用一个条件，限制您在身份池中使用的角色的信任策略。当您为身份池创建或更新角色信任策略时，如果您尝试在没有至少一个限制源身份的条件密钥的情况下保存更改，IAM 会返回错误。Amazon STS 不允许从身份池到缺少此类条件的 IAM 角色进行跨账户AssumeRoleWithWebIdentity操作。

本主题包括几个限制身份池源身份的条件。有关完整列表，请参阅 Amazon Web 联合身份验证的可用密钥。

在使用身份池进行基本身份验证或经典身份验证中， Amazon STS 如果任何 IAM 角色具有正确的信任策略，则可以代入该角色。Amazon Cognito 身份池的 IAM 角色信任服务主体 `cognito-identity.amazonaws.com` 担任该角色。此配置不足以保护您的 IAM 角色来防止意外访问资源。此类角色必须对角色信任策略应用附加条件。如果没有满足以下至少一个条件，您就无法创建或修改身份池的角色。

`cognito-identity.amazonaws.com:aud`

将角色限制为从一个或多个身份池执行操作。Amazon Cognito 在身份池令牌的 `aud` 声明中标明了源身份池。

`cognito-identity.amazonaws.com:amr`

将角色限制为 `authenticated` 或 `unauthenticated`（访客）用户。Amazon Cognito 在身份池令牌的 `amr` 声明中标明了身份验证状态。

`cognito-identity.amazonaws.com:sub`

通过 UUID 将角色限制为一个或多个用户。此 UUID 是用户在身份池中的身份 ID。此值不是来自用户的原始身份提供者的 `sub` 值。Amazon Cognito 在身份池令牌的 `sub` 声明中标明了此 UUID。

增强型流程身份验证要求 IAM 角色与身份池在同一个 Amazon Web Services 账户 ，但在基本身份验证中，情况并非如此。

其他注意事项适用于承担跨账户 IAM 角色的 Amazon Cognito 身份池。这些角色的信任策略必须接受 `cognito-identity.amazonaws.com` 服务主体，并且必须包含特定的 `cognito-identity.amazonaws.com:aud` 条件。为防止意外访问您的 Amazon 资源，aud条件键在条件值中将角色限制为身份池中的用户。

身份池为身份发放的令牌包含有关身份池来源 Amazon Web Services 账户 的信息。当您在 AssumeRoleWithWebIdentityAPI 请求中提供身份池令牌时， Amazon STS 会检查原始身份池是否与 IAM 角色 Amazon Web Services 账户 相同。如果 Amazon STS 确定请求是跨账户的，则它会检查角色信任策略是否有aud条件。如果角色信任策略中不存在此类条件，则 assume-role 调用会失败。如果请求不是跨账户请求，则 Amazon STS 不强制执行此限制。作为一种安全的最佳实践，应始终将此类条件应用于您的身份池角色的信任策略。

其他信任策略条件

跨身份池重复使用角色

要跨多个身份池重复使用某个角色，由于它们共享一个通用权限集，您可以添加多个身份池，如下所示：

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": [
        "us-east-1:12345678-abcd-abcd-abcd-123456790ab",
        "us-east-1:98765432-dcba-dcba-dcba-123456790ab"
    ]
}
```

限制对特定身份的访问权限

要创建限制为一组特定应用用户的策略，请检查 `cognito-identity.amazonaws.com:sub` 的值：

```
"StringEquals": {
    "cognito-identity.amazonaws.com:aud": "us-east-1:12345678-abcd-abcd-
abcd-123456790ab",
    "cognito-identity.amazonaws.com:sub": [
        "us-east-1:12345678-1234-1234-1234-123456790ab",
        "us-east-1:98765432-1234-1234-1243-123456790ab"
    ]
}
```

限制对特定提供商的访问权限

要创建仅限于已使用特定提供商 (可能是您自己的登录提供商) 登录的用户的策略，请检查 `cognito-identity.amazonaws.com:amr` 的值：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "login.myprovider.myapp"
}
```

例如，一个仅信任 Facebook 的应用程序将具有以下 amr 子句：

```
"ForAnyValue:StringLike": {
    "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

访问策略

您附加到某个角色的权限适用于代入该角色的所有用户。为区分用户的访问权限，请使用策略条件和变量。有关更多信息，请参阅 IAM policy 元素：变量和标签。您可以使用该sub条件在访问策略中限制对 Amazon Cognito 身份 IDs 的操作。请谨慎使用此选项，特别是对于未经身份验证的身份，因为这些身份缺少一致的用户 ID。有关使用 Amazon Cognito 进行网络联合的 IAM 策略变量的更多信息，请参阅Amazon Identity and Access Management 用户指南中的 IAM 和 Amazon STS 条件上下文密钥。

为提供更好的安全保护，Amazon Cognito 使用 `GetCredentialsForIdentity`，对在增强型流程中分配给未经身份验证用户的凭证应用缩小范围策略。缩小范围策略可向您对未经身份验证的角色应用的 IAM policy 添加 内联会话策略 和 Amazon 托管会话策略。由于您必须在角色的 IAM policy 和会话策略中授予访问权限，因此，范围缩小策略限制了用户对以下服务列表以外的服务的访问权限。

> **Note**
>
> 在基本（经典）流程中，您可以自己制作 AssumeRoleWithWebIdentityAPI 请求，并且可以将这些限制应用于请求。作为最佳安全实践，请勿向未经身份验证的用户分配超出此缩小范围策略的任何权限。

Amazon Cognito 还可防止经过身份验证和未经身份验证的用户向 Amazon Cognito 身份池和 Amazon Cognito Sync 发出 API 请求。其他人 Amazon Web Services 服务 可能会限制通过 Web 身份访问服务。

在使用增强型流程的成功请求中，Amazon Cognito 在后台发出 `AssumeRoleWithWebIdentity` API 请求。在此请求的参数中，Amazon Cognito 包括以下内容。

1. 用户的身份 ID。
2. 用户所需要代入的 IAM 角色的 ARN。
3. 一个 `policy` 参数，添加内联会话策略。
4. 一个`PolicyArns.member.N`参数，其值为在 Amazon 中授予额外权限的Amazon 托管策略 CloudWatch。

未经身份验证的用户可以访问的服务

当您使用增强型流程时，Amazon Cognito 对您的用户会话应用的范围缩小策略会阻止用户会话使用下表中列出的服务以外的任何服务。对于服务子集，仅允许特定操作。

类别	服务
Analytics	Amazon Data Firehose 适用于 Apache Flink 的亚马逊托管服务
应用程序集成	Amazon Simple Queue Service
AR 和 VR	Amazon Sumerian[1]
业务应用程序	Amazon Mobile Analytics Amazon Simple Email Service
计算	Amazon Lambda
加密和 PKI	Amazon Key Management Service[1]
数据库	Amazon DynamoDB Amazon SimpleDB
前端 Web 和移动	Amazon AppSync Amazon Location Service

类别	服务
	Amazon Simple Notification Service Amazon Pinpoint Amazon Location Service
游戏开发	亚马逊 GameLift 服务器
物联网（IoT）	Amazon IoT
机器学习	Amazon CodeWhisperer Amazon Comprehend Amazon Lex Amazon Machine Learning Amazon Personalize Amazon Polly Amazon Rekognition 亚马逊 SageMaker AI[1] Amazon Textract[1] Amazon Transcribe Amazon Translate
管理与治理	Amazon CloudWatch Amazon CloudWatch 日志
联网和内容分发	Amazon API Gateway
安全性、身份与合规性	Amazon Cognito 用户群体
存储	Amazon Simple Storage Service

[1] 对于下表 Amazon Web Services 服务 中的，内联策略授予操作的子集。该表显示了每个服务中的可用操作。

Amazon Web Services 服务	未经身份验证的增强型流程用户的最大权限

Amazon Key Management Service	`Encrypt`
	`Decrypt`
	`ReEncryptTo`
	`ReEncryptFrom`
	`GenerateDataKey`
	`GenerateDataKeyPair`
	`GenerateDataKeyPair`
	`GenerateDataKeyPairWithoutPlaintext`
	`GenerateDataKeyWithoutPlaintext`

亚马逊 SageMaker AI	`InvokeEndpoint`
Amazon Textract	`DetectDocumentText` `AnalyzeDocument`
Amazon Sumerian	`View*`

Amazon Location Service	`SearchPlaceIndex*`
	`GetPlace`
	`CalculateRoute*`
	`*Geofence`
	`*Geofences`
	`*DevicePosition*`

要向此列表 Amazon Web Services 服务 之外的用户授予访问权限，请在您的身份池中激活基本（经典）身份验证流程。如果您的用户发现分配给未经身份验证的用户的 IAM 角色的策略所允许的`NotAuthorizedException`错误，请评估您是否可以将该服务从您的用例中删除。 Amazon Web Services 服务 如果不能，请切换到基本流程。

访客用户的内联会话策略

Amazon Cognito 首先会在请求 IAM 凭证时应用内联策略。内联会话策略对用户的有效权限进行限制，不能包括对以下列表之外的任何 Amazon Web Services 服务 的访问权限。您还必须在应用于用户的 IAM 角色的策略 Amazon Web Services 服务 中向这些角色授予权限。对于代入角色的会话，用户的有效权限是分配给其角色的策略与其会话策略的交集。有关更多信息，请参阅《Amazon Identity and Access Management 用户指南》中的会话策略。

Amazon Cognito 将以下内联策略添加到原定设置情况下启用的 Amazon Web Services 区域 中的用户会话。若要大概了解内联策略和其他会话策略的最终影响，请参阅未经身份验证的用户可以访问的服务。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "cloudwatch:*",
                "logs:*",
                "dynamodb:*",
                "kinesis:*",
                "mobileanalytics:*",
                "s3:*",
                "ses:*",
                "sns:*",
                "sqs:*",
                "lambda:*",
                "machinelearning:*",
                "execute-api:*",
                "iot:*",
                "gamelift:*",
                "scs:*",
                "cognito-identity:*",
                "cognito-idp:*",
                "lex:*",
                "polly:*",
```

```
                "comprehend:*",
                "translate:*",
                "transcribe:*",
                "rekognition:*",
                "mobiletargeting:*",
                "firehose:*",
                "appsync:*",
                "personalize:*",
                "sagemaker:InvokeEndpoint",
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
            ],
            "Resource": [
                "*"
            ]
        }
    ]
}
```

对于所有其他区域，内联范围缩小策略包括原定设置区域中列出的所有内容，以下 `Action` 语句除外。

```
                "cognito-sync:*",
                "sumerian:View*",
                "codewhisperer:*",
                "textract:DetectDocumentText",
                "textract:AnalyzeDocument",
                "sdb:*"
```

访客 Amazon 托管会话政策

Amazon Cognito 还将 Amazon 托管策略作为会话策略应用于未经身份验证的访客的增强流量会话。此策略通过策略 `AmazonCognitoUnAuthedIdentitiesSessionPolicy` 限制了未经身份验证的用户的权限范围。

您还必须在附加到未经身份验证的 IAM 角色的策略中授予此权限。对于代入角色会话，用户的有效权限是分配给其角色的 IAM 策略与其会话策略的交集。有关更多信息，请参阅《Amazon Identity and Access Management 用户指南》中的会话策略。

有关此 Amazon 托管策略和其他会话策略的净效果的概述，请参阅未经身份验证的用户可以访问的服务。

AmazonCognitoUnAuthedIdentitiesSessionPolicy 托管式策略具有以下权限。

```
{
    "Version": "2012-10-17",
    "Statement": [{
        "Effect": "Allow",
        "Action": [
            "rum:PutRumEvents",
            "polly:*",
            "comprehend:*",
            "translate:*",
            "transcribe:*",
            "rekognition:*",
            "mobiletargeting:*",
            "firehose:*",
            "personalize:*",
            "sagemaker:InvokeEndpoint",
            "geo:GetMap*",
            "geo:SearchPlaceIndex*",
            "geo:GetPlace",
            "geo:CalculateRoute*",
            "geo:*Geofence",
            "geo:*Geofences",
            "geo:*DevicePosition*",
            "kms:Encrypt",
            "kms:Decrypt",
            "kms:ReEncryptTo",
            "kms:ReEncryptFrom",
            "kms:GenerateDataKey",
            "kms:GenerateDataKeyPair",
            "kms:GenerateDataKeyPairWithoutPlaintext",
            "kms:GenerateDataKeyWithoutPlaintext"
        ],
        "Resource": "*"
    }]
}
```

访问策略示例

在本部分中，您可以找到示例 Amazon Cognito 访问策略，这些策略仅向您的用户授予完成特定操作所需的最低权限。您可以在可能的情况下使用策略变量进一步限制给定标识 ID 的权限。例如，使用 ${cognito-identity.amazonaws.com:sub}。有关更多信息，请参阅Amazon 移动博客上的 nderstanding Amazon Cognito Authentication Part 3: Roles and Policies。

> **Note**
>
> 作为安全性最佳实践，策略应仅包括用户执行其任务所需的权限。这意味着您应该尽可能始终为对象限定单个身份的访问范围。

向身份授予对 Amazon S3 中单个对象的读取访问权限

以下访问策略向身份授予读取权限，以便从给定的 S3 存储桶中检索单个对象。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/assets/my_picture.jpg"]
    }
  ]
}
```

向身份授予对 Amazon S3 中身份特定路径的读写访问权限

以下访问策略通过将前缀映射到 `${cognito-identity.amazonaws.com:sub}` 变量来授予读取和写入权限，以访问 S3 存储桶中的特定前缀“文件夹”。

利用此策略，通过 `${cognito-identity.amazonaws.com:sub}` 插入的身份（例如 `us-east-1:12345678-1234-1234-1234-123456790ab`）将能够在 `arn:aws:s3:::amzn-s3-demo-bucket/us-east-1:12345678-1234-1234-1234-123456790ab` 中获取、放置和列出对象。但是，不会授予身份访问 `arn:aws:s3:::amzn-s3-demo-bucket` 中的其他对象的权限。

```
{
```

```
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["s3:ListBucket"],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket"],
      "Condition": {"StringLike": {"s3:prefix": ["${cognito-
identity.amazonaws.com:sub}/*"]}}
    },
    {
      "Action": [
        "s3:GetObject",
        "s3:PutObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::amzn-s3-demo-bucket/${cognito-
identity.amazonaws.com:sub}/*"]
    }
  ]
}
```

Amazon S3 访问权限授予也实现了类似的访问模式。

为 Amazon DynamoDB 分配身份细粒度访问权限

以下访问策略使用 Amazon Cognito 环境变量，为 DynamoDB 资源提供细粒度访问控制。这些变量按身份 ID 授予对 DynamoDB 中项目的访问权限。有关更多信息，请参阅《Amazon DynamoDB 开发人员指南》中的使用 IAM 策略条件实现精细访问控制。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "dynamodb:GetItem",
        "dynamodb:BatchGetItem",
        "dynamodb:Query",
        "dynamodb:PutItem",
        "dynamodb:UpdateItem",
        "dynamodb:DeleteItem",
        "dynamodb:BatchWriteItem"
      ],
```

```
      "Resource": [
        "arn:aws:dynamodb:us-west-2:123456789012:table/MyTable"
      ],
      "Condition": {
        "ForAllValues:StringEquals": {
          "dynamodb:LeadingKeys":  ["${cognito-identity.amazonaws.com:sub}"]
        }
      }
    }
  ]
}
```

向身份授予调用 Lambda 函数的权限

以下访问策略向身份授予调用 Lambda 函数的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
         {
             "Effect": "Allow",
             "Action": "lambda:InvokeFunction",
             "Resource": [
                 "arn:aws:lambda:us-west-2:123456789012:function:MyFunction"
             ]
         }
     ]
 }
```

向身份授予将记录发布到 Kinesis Data Stream 的权限

以下访问策略允许身份将 `PutRecord` 操作与任何 Kinesis Data Streams 结合使用。它可以应用于需要将数据记录添加到账户中所有流的用户。有关更多信息，请参阅《Amazon Kinesis Data Streams 开发人员指南》中的使用 IAM 控制对 Amazon Kinesis Data Streams 资源的访问。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": "kinesis:PutRecord",
            "Resource": [
```

```
                "arn:aws:kinesis:us-east-1:111122223333:stream/stream1"
            ]
        }
    ]
}
```

向身份授予访问 Amazon Cognito 同步存储中其数据的权限

以下访问策略仅向身份授予访问 Amazon Cognito Sync 存储中其自己数据的权限。

```
{
  "Version": "2012-10-17",
  "Statement":[{
      "Effect":"Allow",
      "Action":"cognito-sync:*",
      "Resource":["arn:aws:cognito-sync:us-east-1:123456789012:identitypool/${cognito-
identity.amazonaws.com:aud}/identity/${cognito-identity.amazonaws.com:sub}/*"]
      }]
  }
```

角色信任和权限

这些角色的区别在于其信任关系。下面是未经身份验证角色的示例信任策略：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com.cn:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com.cn:amr": "unauthenticated"
        }
```

```
        }
      }
    ]
}
```

此策略向来自 `cognito-identity.amazonaws.com`（OpenID Connect 令牌的发布者）的联合身份用户授予代入该角色的权限。此外，策略限制令牌的 aud (在此示例中为身份池 ID) 匹配身份池。最后，策略指定的由 Amazon Cognito `GetOpenIdToken` API 操作发布的令牌的多值 `amr` 声明的数组成员之一具有值 `unauthenticated`。

当 Amazon Cognito 创建令牌时，它将令牌的 `amr` 设置为 `unauthenticated` 或 `authenticated`。如果 `amr` 是 `authenticated`，则令牌包括身份验证期间使用的所有提供商。这意味着，您可以创建一个角色，它只信任通过 Facebook 登录的用户，这只需将 `amr` 条件更改为如下所示即可：

```
"ForAnyValue:StringLike": {
  "cognito-identity.amazonaws.com:amr": "graph.facebook.com"
}
```

在更改角色的信任关系或尝试跨身份池使用角色时，请务必谨慎。如果您未正确配置角色来信任身份池，则 STS 结果中会出现类似以下内容的异常：

```
AccessDenied -- Not authorized to perform sts:AssumeRoleWithWebIdentity
```

如果您看到此消息，请仔细检查身份池和身份验证类型是否具有正确的角色。

Amazon Cognito 身份池的安全最佳实践

Amazon Cognito 身份池为您的 Amazon 应用程序提供临时证书。 Amazon Web Services 账户 通常包含应用程序用户所需的资源和私有后端资源。构成 Amazon 证书的 IAM 角色和策略可以授予对其中任何资源的访问权限。

身份池配置的主要最佳实践是确保您的应用程序可以在没有过多权限或意外权限的情况下完成工作。为防止安全配置错误，请在要发布到生产环境的每个应用程序启动之前查看这些建议。

主题

- IAM 配置最佳实践
- 身份池配置最佳实践

IAM 配置最佳实践

当访客或经过身份验证的用户在您的应用程序中启动需要身份池凭证的会话时，您的应用程序会检索 IAM 角色的临时 Amazon 凭证。这些凭证可能是默认角色、由身份池配置中的规则选择的角色或应用程序选择的自定义角色的凭证。将权限分配给每个角色后，您的用户就可以访问您的 Amazon 资源。

有关常规 IAM 最佳实践的更多信息，请参阅 Amazon Identity and Access Management 用户指南中的 IAM 最佳实践。

在 IAM 角色中使用信任策略条件

IAM 要求身份池的角色至少有一个信任策略条件。例如，此条件可以将角色的作用域设置为仅限经过身份验证的用户。 Amazon STS 还要求跨账户基本身份验证请求具有两个特定条件：`cognito-identity.amazonaws.com:aud`和。`cognito-identity.amazonaws.com:amr`作为一项最佳实践，请在信任身份池服务主体 `cognito-identity.amazonaws.com` 的所有 IAM 角色中应用这两个条件。

- `cognito-identity.amazonaws.com:aud`：身份池令牌中的 aud 声明必须与可信身份池 ID 相匹配。
- `cognito-identity.amazonaws.com:amr`：身份池令牌中的 amr 声明必须经过身份验证或未经身份验证。在这个条件下，您可以将角色的访问权限仅限于未经身份验证的访客，或者仅限于经过身份验证的用户。例如，您可以进一步细化此条件的值，将角色限制为来自特定提供者的用户，例如 `graph.facebook.com`。

以下示例角色信任策略在以下条件下授予对角色的访问权限：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com.cn:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-
EXAMPLE11111"
```

```
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com.cn:amr": "authenticated"
        }
      }
    }
  ]
}
```

与身份池相关的元素

- `"Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"`：用户必须来自身份池。
- `"cognito-identity.amazonaws.com.cn:aud": "us-east-1:a1b2c3d4-5678-90ab-cdef-example11111"`：用户必须来自特定身份池 `us-east-1:a1b2c3d4-5678-90ab-cdef-example11111`。
- `"cognito-identity.amazonaws.com.cn:amr": "authenticated"`：用户必须已进行身份验证。访客用户无法代入该角色。

应用最低权限许可

在使用 IAM 策略为经过身份验证的访问或访客访问设置权限时，请仅授予执行特定任务所需的特定权限或最低权限。以下示例 IAM 策略在应用于角色时，授予对 Amazon S3 存储桶中单个映像文件的只读访问权限。

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": [
        "s3:GetObject"
      ],
      "Effect": "Allow",
      "Resource": ["arn:aws:s3:::mybucket/assets/my_picture.jpg"]
    }
  ]
}
```

身份池配置最佳实践

身份池为生成 Amazon 凭证提供了灵活的选项。当您的应用程序可以使用更安全的方法时，不要在设计上采取捷径。

了解访客访问的影响

未经身份验证的访客访问权限允许用户在登录之前从您的 Amazon Web Services 账户 检索数据。知道您的身份池 ID 的任何人都可以请求未经身份验证的凭证。您的身份池 ID 不是机密信息。激活访客访问 Amazon 权限后，所有人均可使用您授予未经身份验证的会话的权限。

妥善的做法是，停用访客访问权限，仅在用户进行身份验证后才获取必需的资源。如果您的应用程序需要在登录前访问资源，请采取以下预防措施。

- 熟悉对未经身份验证的角色设置的自动限制。
- 监控和调整未经身份验证的 IAM 角色的权限，以满足应用程序的特定需求。
- 授予对特定资源的访问权限。
- 保护默认未经身份验证的 IAM 角色的信任策略。
- 只有当您确信自己要将 IAM 角色中的权限授予互联网上的任何人时，才激活访客访问权限。

默认使用增强型身份验证

通过基本（经典）身份验证，Amazon Cognito 会将选择的 IAM 角色委派给您的应用程序。相比之下，增强型流程使用身份池中的集中式逻辑来确定 IAM 角色。它还通过范围缩小策略来设置 IAM 权限上限，为未经验证的身份提供了额外的安全性。增强型流程是更安全的选择，开发人员的工作量更少。要了解有关这些选项的更多信息，请参阅身份池身份验证流程。

基本流程可以公开客户端逻辑，该逻辑用于角色选择和组装 Amazon STS API 凭证请求。增强型流程通过身份池自动化隐藏了角色选择逻辑和代入角色请求。

配置基本身份验证时，请为您的 IAM 角色及其权限应用 IAM 最佳实践。

安全地使用开发人员提供者

经过开发人员验证的身份是服务器端应用程序身份池的一项特征。身份池开发者身份验证所需的唯一身份验证证据是身份池开发者的 Amazon 证书。身份池不会对您在此身份验证流程中提供的开发人员-提供者标识符的有效性施加任何限制。

作为一项最佳实践，仅在以下条件下实施开发人员提供者：

- 为了确保使用经过开发人员验证的凭证的责任能够追溯，请将您的开发人员提供者名称和标识符设计为可指明身份验证来源。例如：`"Logins" : {"MyCorp provider" : "[provider application ID]"}`。
- 避免使用长期用户凭证。配置您的服务器端客户端，以请求具有服务相关角色的身份，例如EC2 实例配置文件和 Lambda 执行角色。
- 避免在同一个身份池中混用内部和外部信任源。在单独的身份池中添加开发人员提供者和单点登录（SSO）提供者。

将属性用于访问控制

访问控制属性是基于属性的访问权限控制（ABAC）的 Amazon Cognito 身份池实现。您可以通过基于用户属性的 Amazon Cognito 身份池使用 IAM 策略来控制对 Amazon 资源的访问。可以从社交和企业身份提供商那里获得这些属性。您可以将提供商的访问权限和 ID 令牌或 SAML 断言中的属性映射到可在 IAM 权限策略中引用的标签。

您可以选择默认映射或在 Amazon Cognito 身份池中创建自己的自定义映射。默认映射让您可以根据一组固定的用户属性编写 IAM 策略。自定义映射允许您选择 IAM 权限策略中引用的一组自定义用户属性。Amazon Cognito 控制台中的 Attribute names (属性名称) 已映射到 Tag key for principal (委托人的标签密钥)，这些是 IAM 权限策略中引用的标签。

例如，假设您拥有一个具有免费和付费会员资格的媒体流式传输服务。您可以将媒体文件存储在 Amazon S3 中，并使用免费或高级标签对其贴标签。您可以将属性用于访问控制，以允许访问基于用户会员级别（这是用户配置文件的一部分）的免费和付费内容。您可以将成员资格属性映射到委托人的标签密钥，以传递给 IAM 权限策略。通过这种方式，您可以创建单个权限策略，并根据会员级别的值和内容文件上的标签有条件地允许对高级内容的访问。

主题

- 使用属性对 Amazon Cognito 身份池进行访问控制
- 示例：将属性用于访问控制策略
- 关闭访问控制属性（控制台）
- 默认提供商映射

使用属性来控制访问有若干优势：

- 使用属性进行访问控制时，权限管理会更高效。您可以创建使用用户属性的基本权限策略，而不必为不同的任务功能创建多个策略。

- 不管您何时为应用程序添加或删除资源或用户，都无需更新策略。权限策略只向具有匹配用户属性的用户授予访问权限。例如，您可能需要根据用户的任务标题控制对某些 S3 存储桶的访问权限。在这种情况下，您可以创建权限策略，以便仅允许定义的任务标题中的用户访问这些文件。有关更多信息，请参阅 IAM 教程：将 SAML 会话标签用于 ABAC。
- 属性可以作为委托人标签传递给策略，该策略基于这些属性的值允许或拒绝权限。

使用属性对 Amazon Cognito 身份池进行访问控制

使用属性进行访问控制之前，请确保满足以下先决条件：

- 一个 Amazon 账户
- 用户池
- 身份池
- 设置 SDK
- 已集成身份提供商
- 凭据

要使用属性进行访问控制，您设置为数据来源的声明将设置您选择的标签键的值。Amazon Cognito 会将标签键和值应用于您的用户会话。您的 IAM policy 可以根据 `${aws:PrincipalTag/tagkey}` 条件评估用户的访问权限。IAM 会根据策略评估用户标签的值。

您必须准备要将其凭证传递给用户的 IAM 角色。这些角色的信任策略必须允许 Amazon Cognito 为您的用户代入该角色。对于用于访问控制的属性，您还必须允许 Amazon Cognito 将主体标签应用于用户的临时会话。授予通过操作代入角色的权限AssumeRoleWithWebIdentity。授予使用仅限权限操作 `sts:TagSession` 标记用户会话的权限。有关更多信息，请参阅《Amazon Identity and Access Management 用户指南》中的在 Amazon Security Token Service中传递会话标签。有关向 Amazon Cognito 服务主体 `cognito-identity.amazonaws.com` 授予 `sts:AssumeRoleWithWebIdentity` 和 `sts:TagSession` 权限的示例信任策略，请参阅示例：将属性用于访问控制策略。

在控制台中配置访问控制属性

1. 登录 Amazon Cognito 控制台并选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。如果要添加新的 IdP，请选择添加身份提供者。

4. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请在访问控制属性中选择编辑。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
5. 选择保存更改。

示例：将属性用于访问控制策略

考虑这样一种场景：某公司法律部门的员工需要列出存储桶中属于其部门并按其安全级别分类的所有文件。假定此员工从身份提供商获得的令牌包含以下陈述。

声明

```
{ .
  .
"sub" : "57e7b692-4f66-480d-98b8-45a6729b4c88",
"department" : "legal",
"clearance" : "confidential",
 .
 .
}
```

这些属性可以映射到标签，并在 IAM 权限策略中作为委托人标签引用。现在，您可以通过更改身份提供商端的用户配置文件来管理访问权限。或者，您可以使用名称或标签来更改资源端的属性，而无需更改策略本身。

以下权限策略有两个作用：

- 允许列表访问以与用户部门名称匹配的前缀结尾的所有 S3 桶。
- 允许对这些存储桶中的文件进行读取访问，只要文件上的清理标签与用户的清理属性匹配。

权限策略

```
    {
     "Version": "2012-10-17",
     "Statement": [
         {
             "Effect": "Allow",
             "Action": "s3:List*",
             "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}"
         },
         {
             "Effect": "Allow",
             "Action": "s3:GetObject*",
             "Resource": "arn:aws:s3:::*-${aws:PrincipalTag/department}/*",
             "Condition": {
                 "StringEquals": {
                     "s3:ExistingObjectTag/clearance": "${aws:PrincipalTag/clearance}"
                 }
             }
         }
     ]
}
```

信任策略决定谁可担任此角色。信任关系策略允许使用 `sts:AssumeRoleWithWebIdentity` 和 `sts:TagSession` 来允许访问。它添加了一部分条件，以将策略限制为您创建的身份池，并确保该策略适用于经身份验证的角色。

信任策略

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"
      },
      "Action": [
        "sts:AssumeRoleWithWebIdentity",
        "sts:TagSession"
      ],
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com.cn:aud": "IDENTITY-POOL-ID"
```

```
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com.cn:amr": "authenticated"
        }
      }
    }
  ]
}
```

关闭访问控制属性（控制台）

按照以下操作步骤来停用访问控制属性。

在控制台中停用访问控制属性

1. 登录 Amazon Cognito 控制台并选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 找到身份提供者。选择要编辑的身份提供者。
4. 在访问控制属性中选择编辑。
5. 如果不应用主体标签，请选择非活动。
6. 选择保存更改。

默认提供商映射

下表提供 Amazon Cognito 支持的身份验证提供商的默认映射信息。

提供商	令牌类型	委托人标签值	示例
Amazon Cognito 用户池	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	"6jk8ltokc7ac9es6jrtg9q572f"，"57e7b692-4f66-480d-98b8-45a6729b4c88"
Facebook	访问令牌	aud（app_id），sub（user_id）	“492844718097981”，“112177216992379”

提供商	令牌类型	委托人标签值	示例
Google	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com”，“109220063452404746097”
SAML	断言	“http://schemas.xmlsoap。 org/ws/2005/05/identity/claims/nameidentifier" , "http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name”	"auth0|5e28d196f8f55a0eaaa95de3"，"user123@gmail.com"
Apple	ID 令牌	aud（客户端 ID）和 sub（用户 ID）	“com.amazonaws.ec2-54-80-172-243.compute-1.client”，“001968.a6ca34e9c1e742458a26cf8005854be9.0733”
Amazon	访问令牌	aud（Amzn Dev Ac 上的客户端 ID），user_id（用户 ID）	“amzn1.application-oa2-client.9d70d9382d3446108aaee3dd763a0fa6”，“amzn1.account。AGHNIFJQMFSBG3G6 XCPVB35ORQAA”

提供商	令牌类型	委托人标签值	示例
标准 OIDC 提供商	ID 令牌和访问令牌	aud（作为 client_id）和 sub（作为用户 ID）	“620493171733-eebk7c0hcp5lj3e1tlqp1gntt3k0rncv.apps.googleusercontent.com”，“109220063452404746097”
Twitter	访问令牌	aud（应用程序 ID；应用程序密钥），sub（用户 ID）	“DfwifTtKEX1Fi IBRnOTI R0CFK；xgj5xb8xir Xg w7fxmwcfvnok9 1y5z1，“1269003884292222976IVCPj” LIdk JJrgwZkLexo
DevAuth	Map	不适用	"tag1", "tag2"

> **Note**
>
> Tag Key for Principal (委托人的标签密钥) 和 Attribute (属性) 名称的默认属性映射选项会自动填充。您无法更改默认映射。

使用基于角色的访问控制

Amazon Cognito 身份池为您的经过身份验证的用户分配一组临时的、权限有限的凭证，以访问您的资源。 Amazon 每个用户的权限通过您创建的 IAM 角色进行控制。您可以定义规则，以便基于用户 ID 令牌中的声明为每个用户选择角色。您可以为经过身份验证的用户定义一个默认角色。您也可以为未经身份验证的来宾用户定义一个具有有限权限的单独的 IAM 角色。

为角色映射创建角色

请务必为每个角色添加适当的信任策略，这样只能由 Amazon Cognito 针对您的身份池中经过身份验证的用户担任。下面是此类信任策略的示例：

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Principal": {
        "Federated": "cognito-identity.cn-north-1.amazonaws.com.cn"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "cognito-identity.amazonaws.com.cn:aud": "us-east-1:12345678-corner-
cafe-123456790ab"
        },
        "ForAnyValue:StringLike": {
          "cognito-identity.amazonaws.com.cn:amr": "authenticated"
        }
      }
    }
  ]
}
```

此策略允许来自 `cognito-identity.amazonaws.com` (OpenID Connect 令牌的发布者) 的联合身份用户担任该角色。此外，策略限制令牌的 `aud` (在此示例中为身份池 ID) 匹配身份池。最后，策略指定的由 Amazon Cognito `GetOpenIdToken` API 操作发布的令牌的多值 `amr` 声明的数组成员之一具有值 `authenticated`。

授予传递角色权限

要允许用户为角色设置超过该用户在身份池上的现有权限的权限，您需授予用户 `iam:PassRole` 权限以将角色传递给 `set-identity-pool-roles` API。例如，如果用户无法写入 Amazon S3，但用户在身份池上设置的 IAM 角色可向 Amazon S3 授予写入权限，则用户只能在已向该角色授予 `iam:PassRole` 权限的情况下设置该角色。以下示例策略介绍了如何提供 `iam:PassRole` 权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "Stmt1",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:PassRole"
            ],
            "Resource": [
                "arn:aws:iam::123456789012:role/myS3WriteAccessRole"
            ]
        }
    ]
}
```

在此策略示例中，将 `iam:PassRole` 权限授予了角色 `myS3WriteAccessRole`。使用角色的 Amazon Resource Name（ARN）指定该角色。您必须将此策略附加到您的用户。有关更多信息，请参阅使用管理的策略。

> **Note**
>
> Lambda 函数使用基于资源的策略，该策略直接附加至 Lambda 函数本身。创建调用 Lambda 函数的规则时，您未传递角色，因此创建规则的用户无需 `iam:PassRole` 权限。有关 Lambda 函数授权的更多信息，请参阅管理权限：使用 Lambda 函数策略。

使用令牌向用户分配角色

对于通过 Amazon Cognito 用户池登录的用户，角色可在用户池分配的 ID 令牌中进行传递。角色将显示在 ID 令牌的以下声明中：

- `cognito:preferred_role` 声明是角色 ARN。
- `cognito:roles`声明是一个以逗号分隔的字符串，其中包含一组允许的角色。 ARNs

声明按如下方式设置：

- `cognito:preferred_role` 声明设置为组中具有最大 (最小) `Precedence` 值的角色。如果只有一个允许的角色，则 `cognito:preferred_role` 设置为该角色。如果存在多个角色且没有一个角色具有最高优先级，则此声明未设置。
- 如果至少存在一个角色，则 `cognito:roles` 声明已设置。

使用令牌分配角色时，如果存在多个可向用户分配的角色，Amazon Cognito 身份池（联合身份）将按以下方式选择角色：

- 如果GetCredentialsForIdentityCustomRoleArn参数已设置并且与cognito:roles声明中的角色匹配，则使用该参数。如果此参数与 cognito:roles 中的角色不匹配，则拒绝访问。
- 如果 cognito:preferred_role 声明已设置，请使用它。
- 如果未设置cognito:preferred_role声明，则cognito:roles声明已设置，CustomRoleArn且未在调用中指定GetCredentialsForIdentity，则使用控制台中的角色解析设置或AmbiguousRoleResolution字段（在 SetIdentityPoolRolesAPI 的RoleMappings参数中）来确定要分配的角色。

使用基于规则的映射向用户分配角色

规则允许您将身份提供商令牌的声明映射到 IAM 角色。

每个规则指定一个令牌声明（例如 Amazon Cognito 用户池的 ID 令牌中的用户属性）、匹配类型、值和 IAM 角色。匹配类型可以是 Equals、NotEqual、StartsWith 或 Contains。如果用户拥有声明的匹配值，则该用户可在获取凭证后担任该角色。例如，您可以创建一个规则，为 custom:dept 自定义属性值为 Sales 的用户分配一个特定的 IAM 角色。

> **Note**
>
> 在规则设置中，自定义属性需要使用 custom: 前缀，以便将它们与标准属性区分开来。

规则按顺序进行评估，并使用第一条匹配规则的 IAM 角色，除非指定 CustomRoleArn 以覆盖顺序。有关 Amazon Cognito 用户池中用户属性的更多信息，请参阅使用用户属性。

您可以在身份池（联合身份）控制台中为身份验证提供商设置多条规则。按顺序应用规则。您可以拖动规则以更改其顺序。第一条匹配规则优先。如果匹配类型为 NotEqual 且声明不存在，则不会对规则进行评估。如果没有规则匹配，则角色解析设置应用到使用经过身份验证的原定设置角色或拒绝请求。

在 API 和 CLI 中，您可以指定在RoleMapping类型字段中没有匹配规则时要分配的角色，该AmbiguousRoleResolution字段在 SetIdentityPoolRolesAPI 的RoleMappings参数中指定。

要在 Amazon Cognito 控制台中向身份提供者添加基于规则的映射，请添加或更新 IdP，然后选择角色选择下面的使用规则选择角色。在这里，您可以添加规则，将提供者声明映射到 IAM 角色。

您可以使用类型的RulesConfiguration字段在 Amazon CLI 或 API 中为身份提供者设置基于规则的RoleMapping映射。您可以在 SetIdentityPoolRolesAPI 的RoleMappings参数中指定此字段。

例如，以下 Amazon CLI 命令添加了一条规则，该规则将角色分配`arn:aws:iam::123456789012:role/Sacramento_team_S3_admin`给萨克拉曼多所在地经过 OIDC IdP 身份验证的用户：`arn:aws:iam::123456789012:oidc-provider/myOIDCIdP`

```
aws cognito-identity set-identity-pool-roles --region us-east-1 --cli-input-json
 file://role-mapping.json
```

`role-mapping.json` 的内容：

```
{
    "IdentityPoolId": "us-east-1:12345678-corner-cafe-123456790ab",
    "Roles": {
        "authenticated": "arn:aws:iam::123456789012:role/myS3WriteAccessRole",
        "unauthenticated": "arn:aws:iam::123456789012:role/myS3ReadAccessRole"
    },
    "RoleMappings": {
        "arn:aws:iam::123456789012:oidc-provider/myOIDCIdP": {
            "Type": "Rules",
            "AmbiguousRoleResolution": "AuthenticatedRole",
            "RulesConfiguration": {
                "Rules": [
                    {
                        "Claim": "locale",
                        "MatchType": "Equals",
                        "Value": "Sacramento",
                        "RoleARN": "arn:aws:iam::123456789012:role/
Sacramento_team_S3_admin"
                    }
                ]
            }
        }
    }
}
```

对于每个用户池或为某个身份池配置的其他身份验证提供商，您可以创建最多 25 条规则。此限制不可调整。有关更多信息，请参阅 Amazon Cognito 中的配额。

基于规则的映射中使用的令牌声明

Amazon Cognito

Amazon Cognito ID 令牌以 JSON Web Token（JWT）表示。令牌包含有关经过身份验证的用户的身份声明，例如 `name`、`family_name` 和 `phone_number`。有关标准声明的更多信息，请参阅 OpenID Connect 规范。除了标准声明之外，以下是特定于 Amazon Cognito 的额外声明：

- `cognito:groups`
- `cognito:roles`
- `cognito:preferred_role`

Amazon

以下声明以及这些声明可能的值可与 Login with Amazon 配合使用：

- `iss`：www.amazon.com
- `aud`：应用程序 ID
- `sub`：Login with Amazon 令牌中的 `sub`

Facebook

以下声明以及这些声明可能的值可与 Facebook 配合使用：

- `iss`：graph.facebook.com
- `aud`：应用程序 ID
- `sub`：Facebook 令牌中的 `sub`

Google

Google 令牌包含 OpenID Connect 规范中的标准声明。OpenID 令牌中的所有声明均可用于基于规则的映射。请参阅 Google 的 OpenID Connect 网站，了解 Google 令牌中包含的声明。

Apple

Apple 令牌包含 OpenID Connect 规范中的标准声明。请参阅 Apple 文档中的使用 Sign in with Apple 对用户进行身份验证，详细了解有关 Apple 令牌提供的声明。Apple 的令牌并不总是包含 `email`。

OpenID

OpenID 令牌中的所有声明均可用于基于规则的映射。有关标准声明的更多信息，请参阅 OpenID Connect 规范。请参阅您的 OpenID 提供商文档，了解其中包含的任何额外声明。

SAML

根据收到的 SAML 断言分析声明。SAML 断言中包含的所有声明均可在基于规则的映射中使用。

基于角色的访问控制的最佳实践

> **Important**
>
> 如果最终用户可修改您映射到角色的声明，则任何最终用户均可担任您的角色并相应地设置策略。仅将无法由最终用户直接设置的声明映射到具有提升的权限的角色。在 Amazon Cognito 用户池中，您可以为每个用户属性设置每个应用程序的读取和写入权限。

> **Important**
>
> 如果您在 Amazon Cognito 用户池中为组设置角色，这些角色将通过用户的 ID 令牌进行传递。要使用这些角色，您还必须为身份池中选择的通过身份验证的角色设置 Choose role from token（使用令牌选择角色）。
> 您可以使用控制台中的角色解析设置和 SetIdentityPoolRolesAPI 的`RoleMappings`参数来指定无法通过令牌确定正确的角色时的默认行为。

获取凭证

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 Amazon 本部分介绍如何获取凭证以及如何从身份池检索 Amazon Cognito 身份。

Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。未经身份验证的用户的身份未经过验证，因此，该角色很适合您的应用程序的来宾用户或用户身份验证与否无关紧要的情形。经过身份验证的用户可以通过第三方身份提供商或证实其身份的用户池登录到您的应用程序。确保您的资源的权限范围适当，让未经身份验证的用户无权访问这些资源。

Amazon Cognito 身份并不是凭证。使用 Amazon Security Token Service (Amazon STS) 中的 Web 联合身份验证支持将它们交换为凭证。建议使用 `AWS.CognitoIdentityCredentials` 来为您的应用程序用户获得 Amazon 凭证。然后使用将凭证对象中的身份交换为证书 Amazon STS。

> **Note**
>
> 如果您的身份池是在 2015 年 2 月前创建的，则必须将角色与身份池重新关联，以便在没有角色作为参数的情况下使用 `AWS.CognitoIdentityCredentials` 构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

Web 身份凭证提供者是中默认凭证提供者链的一部分。 Amazon SDKs要在本地`config`文件中为 Amazon SDK 或设置身份池令牌 Amazon CLI，请添加`web_identity_token_file`个人资料条目。请参阅《工具参考指南》 Amazon SDKs 和《工具参考指南》中的代入角色凭证提供者。

要详细了解如何在 SDK 中填充 Web 身份凭证，请参阅《SDK 开发人员指南》。为了获得最佳效果，请使用内置的身份池集成开始您的项目 Amazon Amplify。

Amazon 用于通过身份池获取和设置凭证的 SDK 资源

- 《Amplify 开发人员中心》的身份池联合身份验证（Android）
- 《Amplify 开发人员中心》的身份池联合身份验证（iOS）
- 开发者指南中@@ 使用亚马逊 Cognito 身份对用户进行 适用于 JavaScript 的 Amazon SDK 身份验证
- 开发者指南中的@@ 亚马逊 Cognito 凭证提供商 适用于 .NET 的 Amazon SDK
- 在《 适用于 Go 的 Amazon SDK 开发者指南》中@@ 以编程方式指定凭证
- 在《 Amazon SDK for Java 2.x 开发人员指南》的代码中提供临时证书
- assumeRoleWithWebIdentityCredentialProvider 适用于 PHP 的 Amazon SDK 开发者指南中的提供商
- 适用于 Python (Boto3) 的 Amazon SDK 文档中的代入 Web 身份提供者的角色
- 在 Amazon SDK for Rust 开发者指南中@@ 指定您的凭证和默认区域

以下各节提供了一些旧版的示例代码 Amazon SDKs。

Android

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 Amazon Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 Amazon 凭证，请按照以下步骤操作。

要在安卓应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的身份验证。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户，则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）。如果您正在对用户进行身份验证，则可以在设置完凭证提供程序中的登录令牌后检索身份 ID：

```
String identityId = credentialsProvider.getIdentityId();
Log.d("LogTag", "my ID is " + identityId);
```

> **Note**
>
> 请勿在应用程序的主线程中调用 `getIdentityId()`、`refresh()` 或 `getCredentials()`。从 Android 3.0（API 级别 11）开始，NetworkOnMainThreadException如果您在主应用程序线程上执行网络 I/O，您的应用程序将自动失败并抛出。您必须使用 `AsyncTask` 将您的代码移至后台线程。有关更多信息，请参阅 Android 文档。您也可以调用 `getCachedIdentityId()` 以检索 ID，但前提是已缓存在本地。否则，该方法将返回 null 值。

iOS - Objective-C

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 Amazon Amazon Cognito 身份池同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 Amazon 凭证，请完成以下步骤。

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证和 Flutter 身份验证。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
// Retrieve your Amazon Cognito ID
[[credentialsProvider getIdentityId] continueWithBlock:^id(AWSTask *task) {
    if (task.error) {
        NSLog(@"Error: %@", task.error);
```

```
    }
    else {
        // the task result will contain the identity id
        NSString *cognitoId = task.result;
    }
    return nil;
}];
```

> **Note**
>
> `getIdentityId` 为异步调用。如果您已在提供程序上设置身份 ID，则可以调用 `credentialsProvider.identityId` 以检索已缓存在本地的身份。但是，如果您未在提供程序上设置身份 ID，则调用 `credentialsProvider.identityId` 将返回 `nil`。有关更多信息，请参阅 Amplify iOS SDK 参考。

iOS - Swift

您可以使用 Amazon Cognito 为您的应用程序提供临时的、有限权限的证书，以便您的用户可以访问资源。 Amazon Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 Amazon 凭证，请按照以下步骤操作。

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
// Retrieve your Amazon Cognito ID
credentialsProvider.getIdentityId().continueWith(block: { (task) -> AnyObject? in
    if (task.error != nil) {
        print("Error: " + task.error!.localizedDescription)
    }
    else {
        // the task result will contain the identity id
        let cognitoId = task.result!
        print("Cognito id: \(cognitoId)")
    }
    return task;
```

```
})
```

> **Note**
>
> `getIdentityId` 为异步调用。如果您已在提供程序上设置身份 ID，则可以调用 `credentialsProvider.identityId` 以检索已缓存在本地的身份。但是，如果您未在提供程序上设置身份 ID，则调用 `credentialsProvider.identityId` 将返回 `nil`。有关更多信息，请参阅 Amplify iOS SDK 参考。

JavaScript

如果您尚未创建身份池，则在使用 `AWS.CognitoIdentityCredentials` 之前，先在 Amazon Cognito 控制台中创建一个身份池。

通过您的身份提供商配置身份池后，您可以使用 `AWS.CognitoIdentityCredentials` 验证用户身份。要将应用程序凭证配置为使用 `AWS.CognitoIdentityCredentials`，则为 `credentials` 或基于每个服务配置设置 `AWS.Config` 属性。以下示例使用 `AWS.Config`：

```
// Set the region where your identity pool exists (us-east-1, eu-west-1)
AWS.config.region = 'us-east-1';

// Configure the credentials provider to use your identity pool
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'IDENTITY_POOL_ID',
    Logins: { // optional tokens, used for authenticated login
        'graph.facebook.com': 'FBTOKEN',
        'www.amazon.com': 'AMAZONTOKEN',
        'accounts.google.com': 'GOOGLETOKEN',
        'appleid.apple.com': 'APPLETOKEN'
    }
});

// Make the call to obtain credentials
AWS.config.credentials.get(function(){

    // Credentials will be available when this function is called.
    var accessKeyId = AWS.config.credentials.accessKeyId;
    var secretAccessKey = AWS.config.credentials.secretAccessKey;
    var sessionToken = AWS.config.credentials.sessionToken;
```

```
});
```

可选的 `Logins` 属性是身份提供商名称到这些提供商身份令牌的映射。您如何从身份提供商获得令牌的方式取决于您使用的提供商。例如，如果 Facebook 是您的身份提供商之一，则您可以使用来自 Facebook 软件开发工具包的 `FB.login` 函数获取身份提供商令牌：

```
FB.login(function (response) {
    if (response.authResponse) { // logged in
        AWS.config.credentials = new AWS.CognitoIdentityCredentials({
          IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030',
          Logins: {
            'graph.facebook.com': response.authResponse.accessToken
          }
        });

        console.log('You are now logged in.');
    } else {
        console.log('There was a problem logging you in.');
    }
});
```

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
var identityId = AWS.config.credentials.identityId;
```

Unity

您可以使用 Amazon Cognito 为您的应用程序提供临时的、权限有限的证书，以便您的用户可以访问资源。 Amazon Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 Amazon 凭证，请按照以下步骤操作。

适用于 Unity 的Amazon SDK 现在是 适用于 .NET 的 Amazon SDK 的一部分。要开始使用中的亚马逊 Cognito 适用于 .NET 的 Amazon SDK，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。适用于 .NET 的 Amazon SDK 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。Amazon Amplify

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
credentials.GetIdentityIdAsync(delegate(AmazonCognitoIdentityResult<string> result) {
    if (result.Exception != null) {
        //Exception!
    }
    string identityId = result.Response;
});
```

Xamarin

您可以使用 Amazon Cognito 为您的应用程序提供临时的、有限权限的证书，以便您的用户可以访问资源。 Amazon Amazon Cognito 同时支持经过身份验证和未经身份验证的身份。要为您的应用程序提供 Amazon 凭证，请按照以下步骤操作。

适用于 Xamarin 的Amazon SDK 现在是 适用于 .NET 的 Amazon SDK 的一部分。要开始使用中的亚马逊 Cognito 适用于 .NET 的 Amazon SDK，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。适用于 .NET 的 Amazon SDK 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。Amazon Amplify

> **Note**
>
> 注意：如果您在 2015 年 2 月之前创建了身份池，您必须将您的角色与身份池重新关联，才能在没有角色作为参数的情况下使用此构造函数。为此，请打开 Amazon Cognito 控制台，选择 Manage identity pools（管理身份池）、选择您的身份池，然后选择 Edit identity Pool（编辑身份池），指定您的经过身份验证的角色和未经身份验证的角色，然后保存更改。

检索 Amazon Cognito 身份

如果您允许未经身份验证的用户或者已设置完凭证提供程序中的登录令牌（如果您正在对用户进行身份验证），则可以立即检索终端用户的唯一 Amazon Cognito 标识符（身份 ID）：

```
var identityId = await credentials.GetIdentityIdAsync();
```

使用临时 Amazon Web Services 服务 凭证进行访问

使用身份池成功进行身份验证的结果是一组 Amazon 凭证。使用这些证书，您的应用程序可以向受 IAM 身份验证保护的 Amazon 资源发出请求。借助您可以添加到应用程序中以访问身份池 API 操作的各种 Amazon SDKs 功能，您可以发出未经身份验证的 API 请求，从而生成临时证书。然后，您可以将其他 SDKs 内容 Amazon Web Services 服务 添加到您的客户端，并使用这些临时证书签署请求。向您的临时凭证角色授予的 IAM 权限必须允许您从其他服务请求的操作。

在配置您的 Amazon Cognito 凭证提供程序并检索 Amazon 证书后，创建一个 Amazon Web Services 服务 客户端。以下是 Amazon SDK 文档中的一些示例。

Amazon 用于创建客户端的 SDK 资源

- Amazon 《适用于 C++ 的 Amazon SDK 开发人员指南》中的@@ 客户端配置
- 将 适用于 Go 的 Amazon SDK V2 与《适用于 Go 的 Amazon SDK 开发人员指南》 Amazon Web Services 服务中的搭配使用
- 在《Amazon SDK for Java 2.x 开发人员指南》中@@ 配置 HTTP 客户端
- 在《适用于 JavaScript 的 Amazon SDK 开发者指南》中@@ 创建和调用服务对象
- 在 适用于 Python (Boto3) 的 Amazon SDK 文档中@@ 创建客户端
- 在《Amazon SDK for Rust 开发者指南》中@@ 创建服务客户端
- 在《Amazon SDK for Swift 开发者指南》中@@ 使用客户端

下面的代码段初始化 Amazon DynamoDB 客户端：

Android

要在安卓应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的身份验证。

```
// Create a service client with the provider
AmazonDynamoDB client = new AmazonDynamoDBClient(credentialsProvider);
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 Amazon 权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

iOS - Objective-C

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证和 Flutter 身份验证。

```
// create a configuration that uses the provider
AWSServiceConfiguration *configuration = [AWSServiceConfiguration
 configurationWithRegion:AWSRegionUSEast1 provider:credentialsProvider];
// get a client with the default service configuration
AWSDynamoDB *dynamoDB = [AWSDynamoDB defaultDynamoDB];
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 Amazon 权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

iOS - Swift

要在 iOS 应用程序中使用 Amazon Cognito 身份池，请进行设置。 Amazon Amplify有关更多信息，请参阅《Amplify 开发中心》中的 Swift 身份验证。

```
// get a client with the default service configuration
let dynamoDB = AWSDynamoDB.default()

// get a client with a custom configuration
AWSDynamoDB.register(with: configuration!, forKey: "USWest2DynamoDB");
let dynamoDBCustom = AWSDynamoDB(forKey: "USWest2DynamoDB")
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时有限 Amazon 权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

JavaScript

```
// Create a service client with the provider
var dynamodb = new AWS.DynamoDB({region: 'us-west-2'});
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 Amazon 有限权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

Unity

适用于 Unity 的Amazon SDK 现在是 适用于 .NET 的 Amazon SDK 的一部分。要开始使用中的亚马逊 Cognito 适用于 .NET 的 Amazon SDK，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。适用于 .NET 的 Amazon SDK 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。Amazon Amplify

```
// create a service client that uses credentials provided by Cognito
AmazonDynamoDBClient client = new AmazonDynamoDBClient(credentials, REGION);
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 Amazon 有限权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

Xamarin

适用于 Xamarin 的Amazon SDK 现在是 适用于 .NET 的 Amazon SDK 的一部分。要开始使用中的亚马逊 Cognito 适用于 .NET 的 Amazon SDK，请参阅开发者指南中的亚马逊 Cognito 凭证提供商。适用于 .NET 的 Amazon SDK 或者，请参阅 Amplify 开发者中心，了解用于构建应用程序的选项。Amazon Amplify

```
// create a service client that uses credentials provided by Cognito
var client = new AmazonDynamoDBClient(credentials, REGION)
```

凭证提供者与 Amazon Cognito 通信，检索经过身份验证和未经身份验证的用户的唯一标识符，以及移动 SDK 的临时 Amazon 有限权限证书。 Amazon 检索到的凭证的有效期为 1 小时，提供程序会在凭证过期时进行刷新。

身份池第三方身份提供者

借助 Amazon Cognito 身份池，您可以与各种外部身份提供商 (IdPs) 集成，通过应用程序中的联合身份验证提供临时 Amazon 证书。通过将您的身份池配置为使用这些外部 Amazon 资源 IdPs，您可以通过 Amazon Cognito 用户池、社交提供商、OIDC 提供商或 SAML 提供商进行身份验证，授权您的用户访问后端资源。本节介绍设置和集成 IdPs Amazon Cognito 身份池的步骤。

使用 `logins` 属性，您可以设置从身份提供商 (IdP) 处收到的凭证。您也可以将一个身份池与多个身份池相关联 IdPs。例如，您可以在 `logins` 属性中设置 Facebook 和 Google 令牌，以将唯一的

Amazon Cognito 身份与两个 IdP 登录相关联。用户可以使用任一账户进行身份验证，但 Amazon Cognito 返回同一用户标识符。

以下说明将指导您使用 Amazon Cognito 身份池支持的身份池进行身份验证。 IdPs

主题

- 将 Facebook 设置为身份池 IdP
- 设置 Login with Amazon 作为身份池 IdP
- 将 Google 设置为身份池 IdP
- 使用 Apple 作为身份池 IdP 来设置登录
- 设置 OIDC 提供者作为身份池 IdP
- 设置 SAML 提供者作为身份池 IdP

将 Facebook 设置为身份池 IdP

Amazon Cognito 身份池与 Facebook 配合使用，以便针对应用程序用户提供联合身份验证。本部分介绍如何使用 Facebook 作为 IdP 来注册和设置应用程序。

设置 Facebook

在对 Facebook 用户进行身份验证并与 Facebook 互动之前，请先在 Facebook APIs 上注册

Facebook 开发人员门户可帮助您设置应用程序。在将 Facebook 集成到您的 Amazon Cognito 身份池之前，请执行以下过程：

> **Note**
>
> Amazon Cognito 身份池联合身份验证与 Facebook Limited Login 不兼容。有关如何在不超过 Limited Login 权限集的情况下设置 Facebook Login for iOS 的更多信息，请参阅 Meta for Developers 中的 Facebook Login for iOS - 快速入门。

设置 Facebook

1. 在 Facebook 开发人员门户中，使用 Facebook 凭证登录。
2. 从 Apps（应用程序）菜单中，选择 Add a New App（添加新应用程序）。
3. 选择一个平台，然后完成快速启动流程。

Android

有关如何将 Android 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

iOS - Objective-C

有关如何将 OS Objective-C 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

iOS - Swift

有关如何将 iOS Swift 应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

JavaScript

有关如何将 JavaScript 网络应用程序与 Facebook 登录集成的更多信息，请参阅 Facebook 入门指南。

在 Amazon Cognito 身份池控制台中配置身份提供者

使用以下过程配置身份提供者。

添加 Facebook 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Facebook。
5. 输入你在 Meta 开发者版中创建的 OAuth 项目的应用程序 ID。有关更多信息，请参阅《Meta for Developers 文档》中的 Facebook 登录。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。

7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

使用 Facebook

Android

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，将 Login with Facebook（使用 Facebook 登录）按钮添加到 Android 用户界面。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序中。

Facebook SDK 4.0 或更高版本：

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", AccessToken.getCurrentAccessToken().getToken());
credentialsProvider.setLogins(logins);
```

Facebook SDK 4.0 之前的版本：

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("graph.facebook.com", Session.getActiveSession().getAccessToken());
credentialsProvider.setLogins(logins);
```

Facebook 登录流程在其开发工具包中初始化一个单例会话。Facebook 会话对象包含一个 OAuth 令牌，Amazon Cognito 使用该令牌为经过身份验证的最终用户生成 Amazon 证书。Amazon Cognito 还使用该令牌检查用户数据库，以确定是否存在与此特定 Facebook 身份匹配的用户。如果用户已存在，则 API 会返回现有的标识符。否则，API 将返回新的标识符。客户端开发工具包会自动在本地设备上缓存标识符。

> **Note**
>
> 设置登录映射后，调用`refresh`或`get`检索 Amazon 证书。

iOS - Objective-C

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，向用户界面添加“用 Facebook 登录”按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，并将其绑定到唯一的 Amazon Cognito 身份池（联合身份）。

要向亚马逊 Cognito 提供 Facebook 访问令牌，请实施 AWSIdentityProviderManager协议。

在实施 `logins` 方法时，返回一个包含 `AWSIdentityProviderFacebook` 的词典。此词典充当键，而经过身份验证的 Facebook 用户的当前访问令牌充当值，如以下代码示例所示。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    FBSDKAccessToken* fbToken = [FBSDKAccessToken currentAccessToken];
    if(fbToken){
        NSString *token = fbToken.tokenString;
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook : token }];
    }else{
            return [AWSTask taskWithError:[NSError errorWithDomain:@"Facebook Login"
                                                              code:-1
                                                          userInfo:@{@"error":@"No current
 Facebook access token"}]];
    }
}
```

当您实例化 `AWSCognitoCredentialsProvider` 时，在构造函数中传递实施 `AWSIdentityProviderManager` 作为 `identityProviderManager` 的值的类。有关更多信息，请转到AWSCognitoCredentialsProvider参考页面并选择initWithRegion类型:identityPoolId: identityProviderManager。

iOS - Swift

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。然后，向用户界面添加“用 Facebook 登录”按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，并将其绑定到唯一的 Amazon Cognito 身份池（联合身份）。

> **Note**
>
> Amazon Cognito 身份池联合身份验证与 Facebook Limited Login 不兼容。有关如何在不超过 Limited Login 权限集的情况下设置 Facebook Login for iOS 的更多信息，请参阅 Meta for Developers 中的 Facebook Login for iOS - 快速入门。

要向亚马逊 Cognito 提供 Facebook 访问令牌，请实施 AWSIdentityProviderManager协议。

在实施 `logins` 方法时，返回一个包含 `AWSIdentityProviderFacebook` 的词典。此词典充当键，而经过身份验证的 Facebook 用户的当前访问令牌充当值，如以下代码示例所示。

```
class FacebookProvider: NSObject, AWSIdentityProviderManager {
    func logins() -> AWSTask<NSDictionary> {
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }
}
```

当您实例化 `AWSCognitoCredentialsProvider` 时，在构造函数中传递实施 `AWSIdentityProviderManager` 作为 `identityProviderManager` 的值的类。欲了解更多信息，请访问 AWSCognitoCredentialsProvider参考页面并选择initWithRegion类型:identityPoolId: identityProviderManager。

JavaScript

要添加 Facebook 身份验证，请按照适用于 Web 的 Facebook 登录中的说明操作，并在您的网站上添加 Login with Facebook （使用 Facebook 登录）按钮。Facebook 开发工具包使用会话对象跟踪其状态。Amazon Cognito 使用来自此会话对象的访问令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序中。

```
FB.login(function (response) {

  // Check if the user logged in successfully.
  if (response.authResponse) {
```

```
    console.log('You are now logged in.');

    // Add the Facebook access token to the Amazon Cognito credentials login map.
    AWS.config.credentials = new AWS.CognitoIdentityCredentials({
      IdentityPoolId: 'IDENTITY_POOL_ID',
      Logins: {
        'graph.facebook.com': response.authResponse.accessToken
      }
    });

    // Obtain AWS credentials
    AWS.config.credentials.get(function(){
        // Access AWS resources here.
    });

  } else {
    console.log('There was a problem logging you in.');
  }

});
```

Facebook 软件开发工具包会获取一个 OAuth 令牌，亚马逊 Cognito 使用该令牌为经过身份验证的最终用户 Amazon 生成证书。Amazon Cognito 还可以使用该令牌检查用户数据库，以确定是否存在与此特定 Facebook 身份匹配的用户。如果用户已存在，则 API 会返回现有的标识符。否则，将返回新的标识符。标识符由本地设备上的客户端开发工具包自动缓存。

> **Note**
>
> 设置登录映射后，请调用 `refresh` 或 `get` 以获取凭证。有关代码示例，请参阅JavaScript 自述文件中的 “用例 17，将用户池与 Cognito 身份集成”。

Unity

要添加 Facebook 身份验证，请先按照 Facebook 指南将 Facebook 开发工具包集成到应用程序中。Amazon Cognito 使用 FB 对象中的 Facebook 访问令牌生成与 Amazon Cognito 身份关联的唯一用户标识符。

使用 Facebook 开发工具包对用户进行身份验证后，将会话令牌添加到 Amazon Cognito 凭证提供程序：

```
void Start()
{
    FB.Init(delegate() {
        if (FB.IsLoggedIn) { //User already logged in from a previous session
            AddFacebookTokenToCognito();
        } else {
            FB.Login ("email", FacebookLoginCallback);
        }
    });
}

void FacebookLoginCallback(FBResult result)
{
    if (FB.IsLoggedIn)
    {
        AddFacebookTokenToCognito();
    }
    else
    {
        Debug.Log("FB Login error");
    }
}

void AddFacebookTokenToCognito()
{
    credentials.AddLogin ("graph.facebook.com",
 AccessToken.CurrentAccessToken.TokenString);
}
```

使用 `FB.AccessToken` 之前，调用 `FB.Login()` 并确保 `FB.IsLoggedIn` 为 True。

Xamarin

Xamarin for Android：

```
public void InitializeFacebook() {
    FacebookSdk.SdkInitialize(this.ApplicationContext);
    callbackManager = CallbackManagerFactory.Create();
    LoginManager.Instance.RegisterCallback(callbackManager, new FacebookCallback &lt;
 LoginResult &gt; () {
      HandleSuccess = loginResult = &gt; {
        var accessToken = loginResult.AccessToken;
        credentials.AddLogin("graph.facebook.com", accessToken.Token);
```

```
        //open new activity
      },
      HandleCancel = () = &gt; {
        //throw error message
      },
      HandleError = loginError = &gt; {
        //throw error message
      }
    });
    LoginManager.Instance.LogInWithReadPermissions(this, new List &lt; string &gt; {
      "public_profile"
    });
  }
```

Xamarin for iOS：

```
public void InitializeFacebook() {
  LoginManager login = new LoginManager();
  login.LogInWithReadPermissions(readPermissions.ToArray(),
 delegate(LoginManagerLoginResult result, NSError error) {
    if (error != null) {
      //throw error message
    } else if (result.IsCancelled) {
      //throw error message
    } else {
      var accessToken = loginResult.AccessToken;
      credentials.AddLogin("graph.facebook.com", accessToken.Token);
      //open new view controller
    }
  });
}
```

设置 Login with Amazon 作为身份池 IdP

Amazon Cognito 身份池与 Login with Amazon 配合使用，以便针对移动应用程序和 Web 应用程序用户提供联合身份验证。本部分介绍如何使用 Login with Amazon（使用亚马逊账号登录）作为身份提供商 (IdP) 来注册和设置应用程序。

在开发人员门户中设置 Login with Amazon（使用亚马逊账号登录）用于 Amazon Cognito。有关更多信息，请参阅 Login with Amazon（使用亚马逊账号登录）常见问题中的设置 Login with Amazon（使用亚马逊账号登录）。

> **Note**
>
> 要将 Login with Amazon（使用亚马逊账号登录）集成到 Xamarin 应用程序中，请按照 Xamarin 入门指南中的说明操作。

> **Note**
>
> 您无法在 Unity 平台上原生集成 Login with Amazon（使用亚马逊账号登录）。请使用 Web 视图并完成浏览器登录流程。

设置 Login with Amazon（使用亚马逊账号登录）

实施 Login with Amazon（使用亚马逊账号登录）

在亚马逊开发者门户中，您可以设置 OAuth 应用程序以与您的身份池集成，查找 Login with Amazon 文档，然后下载 SDKs。选择 Developer console（开发人员控制台），然后在开发人员门户中选择 Login with Amazon（使用亚马逊账号登录）。您可以为您的应用程序创建一个安全配置文件，然后在您的应用程序中构建 Login with Amazon（使用亚马逊账号登录）身份验证机制。请参阅获取凭证，了解有关如何将 Login with Amazon（使用亚马逊账号登录）身份验证与应用程序集成的更多信息

Amazon 会为您的新安全配置文件发放 OAuth 2.0 客户端 ID。您可以在安全配置文件的 Web Settings（Web 设置）选项卡上找到 client ID（客户端 ID）。在您身份池中的“通过 Amazon 登录”IdP 的应用程序 ID 字段中输入安全配置文件 ID。

> **Note**
>
> 在您身份池中的“通过 Amazon 登录”IdP 的应用程序 ID 字段中输入安全配置文件 ID。这与使用客户端 ID 的用户群体不同。

在 Amazon Cognito 控制台中配置外部提供商

添加 Login with Amazon 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。

3. 选择添加身份提供者。
4. 选择 Login with Amazon。
5. 输入您在 Lo gin with Amazon 上创建的 OAuth 项目的应用程序 ID。有关更多信息，请参阅 Login with Amazon 文档。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

使用 Login with Amazon：Android

对亚马逊登录进行身份验证后，您可以通过界面的 onSuccess 方法将令牌传递给 Amazon Cognito 凭证提供商。 TokenListener 代码如下所示：

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString(AuthzConstants.BUNDLE_KEY.TOKEN.val);
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("www.amazon.com", token);
    credentialsProvider.setLogins(logins);
}
```

使用 Login with Amazon：iOS - Objective-C

对亚马逊登录进行身份验证后，您可以通过以下 requestDidSucceed 方法将令牌传递给 Amazon Cognito 凭证提供商： AMZNAccessTokenDelegate

```
- (void)requestDidSucceed:(APIResult \*)apiResult {
    if (apiResult.api == kAPIAuthorizeUser) {
        [AIMobileLib getAccessTokenForScopes:[NSArray arrayWithObject:@"profile"]
 withOverrideParams:nil delegate:self];
    }
    else if (apiResult.api == kAPIGetAccessToken) {
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyLoginWithAmazon):
 apiResult.result };
    }
}}
```

使用 Login with Amazon：iOS - Swift

在对 Amazon 登录进行身份验证后，您可以在 `AMZNAccessTokenDelegate` 的 `requestDidSucceed` 方法中将令牌传递给 Amazon Cognito 凭证提供程序：

```
func requestDidSucceed(apiResult: APIResult!) {
    if apiResult.api == API.AuthorizeUser {
        AIMobileLib.getAccessTokenForScopes(["profile"], withOverrideParams: nil,
 delegate: self)
    } else if apiResult.api == API.GetAccessToken {
        credentialsProvider.logins =
 [AWSCognitoLoginProviderKey.LoginWithAmazon.rawValue: apiResult.result]
    }
}
```

使用 Login with Amazon： JavaScript

在用户通过 Login with Amazon 进行身份验证并重定向回网站后，系统会在查询字符串中提供 Login with Amazon access_token。将此令牌传递到凭证登录映射。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   Logins: {
       'www.amazon.com': 'Amazon Access Token'
   }
```

```
});
```

将 Google 设置为身份池 IdP

Amazon Cognito 身份池与 Google 配合使用，以便针对移动应用程序用户提供联合身份验证。本部分介绍如何使用 Google 作为 IdP 来注册和设置应用程序。

Android

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商。将所有创建的客户添加 IDs 为其他受众值，以实现更好的集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要激活 Google Sign-in for Android，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择APIs 和服务，然后选择OAuth 同意屏幕。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择OAuth 客户端 ID。选择 Android 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关如何将 Google 集成到 Android 应用程序的更多信息，请参阅 Google Identity 文档中的使用 Sign in with Google 验证用户身份。

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在谷歌云平台上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅 Google 云平台控制台帮助中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

使用 Google

要在应用程序中启用 Login with Google，请按照适用于 Android 的 Google 文档中的说明执行操作。当用户登录时，他们向 Google 请求 OpenID Connect 身份验证令牌。然后，Amazon Cognito 将使用该令牌对用户进行身份验证并生成一个唯一标识符。

以下示例代码显示如何从 Google Play 服务检索身份验证令牌：

```
GooglePlayServicesUtil.isGooglePlayServicesAvailable(getApplicationContext());
```

```
AccountManager am = AccountManager.get(this);
Account[] accounts = am.getAccountsByType(GoogleAuthUtil.GOOGLE_ACCOUNT_TYPE);
String token = GoogleAuthUtil.getToken(getApplicationContext(), accounts[0].name,
        "audience:server:client_id:YOUR_GOOGLE_CLIENT_ID");
Map<String, String> logins = new HashMap<String, String>();
logins.put("accounts.google.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商。将所有创建的客户添加 IDs 为其他受众值，以实现更好的集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要启用 Google Sign-in for iOS，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择APIs 和服务，然后选择OAuth 同意屏幕。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择OAuth 客户端 ID。选择 iOS 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户。选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关将 Google 集成到 iOS 应用程序的更多信息，请参阅 Google Identity 文档中的 Google Sign-In for iOS。

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在谷歌云平台上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅 Google 云平台控制台帮助中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 iOS 的 Google 文档中的说明执行操作。成功通过身份验证后，将生成一个 OpenID Connect 身份验证令牌，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符。

成功通过身份验证后，将生成一个 `GTMOAuth2Authentication` 对象，其中包含一个 `id_token`，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符：

```
- (void)finishedWithAuth: (GTMOAuth2Authentication *)auth error: (NSError *) error {
        NSString *idToken = [auth.parameters objectForKey:@"id_token"];
        credentialsProvider.logins = @{ @(AWSCognitoLoginProviderKeyGoogle): idToken };
    }
```

iOS - Swift

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商。将所有创建的客户添加 IDs 为其他受众值，以实现更好的集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要启用 Google Sign-in for iOS，请为应用程序创建 Google Developers 控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择APIs 和服务，然后选择OAuth 同意屏幕。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择OAuth 客户端 ID。选择 iOS 作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关将 Google 集成到 iOS 应用程序的更多信息，请参阅 Google Identity 文档中的 Google Sign-In for iOS。

在 Amazon Cognito 控制台主页中选择 Manage Identity Pools（管理身份池）：

在 Amazon Cognito 控制台中配置外部提供商

1. 选择想要在其中启用 Google 作为外部提供商的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在控制面板页面的右上角，选择 Edit identity pool (编辑身份池)。此时将显示“Edit identity pool”（编辑身份池）页。
3. 向下滚动并选择 Authentication providers（身份验证提供商）以展开这一部分。
4. 选择 Google 选项卡。
5. 选择 Unlock（解锁）。
6. 输入从 Google 获取的 Google 客户端 ID，然后选择 Save Changes（保存更改）。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 iOS 的 Google 文档中的说明执行操作。成功通过身份验证后，将生成一个 OpenID Connect 身份验证令牌，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符。

成功的身份验证会生成一个包含 `id_token` 的 `GTMOAuth2Authentication` 对象。Amazon Cognito 使用此令牌对用户进行身份验证并生成一个唯一标识符：

```
func finishedWithAuth(auth: GTMOAuth2Authentication!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.parameters.objectForKey("id_token")
      credentialsProvider.logins = [AWSCognitoLoginProviderKey.Google.rawValue:
 idToken!]
    }
}
```

JavaScript

> **Note**
>
> 如果您的应用程序使用 Google 并且可在多个移动平台上使用，则应将 Google 配置为 OpenID Connect 提供商。将所有创建的客户添加 IDs 为其他受众值，以实现更好的集成。要了解有关 Google 跨客户端身份模式的更多信息，请参阅跨客户端身份。

设置 Google

要为 JavaScript 网络应用程序启用 Google 登录，请为您的应用程序创建 Google 开发者控制台项目。

1. 转到 Google Developers 控制台并创建一个新项目。
2. 选择APIs 和服务，然后选择OAuth 同意屏幕。自定义 Google 在征求用户同意以便与您的应用共享个人资料数据时向用户显示的信息。
3. 选择 Credentials（凭证），然后选择 Create credentials（创建凭证）。选择OAuth 客户端 ID。选择 Web application（Web 应用程序）作为 Application type（应用程序类型）。为开发应用程序的每个平台创建单独的客户端 ID。
4. 从 Credentials（凭证）中，选择 Manage service accounts（管理服务账户）。选择 Create service account（创建服务账户）。输入服务账户详细信息，然后选择 Create and continue（创建并继续）。
5. 授予服务账户对项目的访问权限。根据应用程序的要求授予用户访问服务账户的权限。
6. 选择您的新服务账户，选择 Keys（密钥）选项卡，然后选择 Add key（添加密钥）。创建并下载新的 JSON 密钥。

有关如何使用 Google Developers 控制台的更多信息，请参阅 Google Cloud 文档中的创建和管理项目。

有关如何将 Google 集成到 Web 应用程序的更多信息，请参阅 Google Identity 文档中的使用 Google 登录。

在 Amazon Cognito 控制台中配置外部提供商

添加 Google 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。

2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 Google。
5. 输入您在谷歌云平台上创建的 OAuth 项目的客户端 ID。有关更多信息，请参阅 Google 云平台控制台帮助中的设置 OAuth 2.0。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 `sub` 和 `aud` 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

使用 Google

要在应用程序中启用“用 Google 登录”，请按照适用于 Web 的 Google 文档中的说明执行操作。

成功通过身份验证后，将生成一个响应对象，其中包含一个 `id_token`，供 Amazon Cognito 用于对用户进行身份验证并生成一个唯一标识符：

```
function signinCallback(authResult) {
  if (authResult['status']['signed_in']) {

     // Add the Google access token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
```

```
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'accounts.google.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
     AWS.config.credentials.get(function(){
        // Access AWS resources here.
     });
  }
}
```

使用 Apple 作为身份池 IdP 来设置登录

Amazon Cognito 身份池与“通过 Apple 登录”配合使用，以便针对移动应用程序和 Web 应用程序用户提供联合身份验证。本节介绍如何使用 Sign in with Apple（使用苹果账号登录）作为身份提供商 (IdP) 来注册和设置应用程序。

要将 Sign in with Apple 作为身份验证提供商添加到身份池，您必须完成两个过程。首先，在应用程序中集成 Sign in with Apple（使用苹果账号登录），然后在身份池中配置 Sign in with Apple（使用苹果账号登录）。有关设置 “使用 Apple 登录” 的 up-to-date更多信息，请参阅 Apple 开发者文档中的配置环境以使用 Apple 登录。

设置 Sign in with Apple

要将 Sign in with Apple（使用苹果账号登录）配置为 IdP，您必须向 Apple 注册您的应用程序才能接收客户端 ID。

1. 创建 Apple 开发人员账户。
2. 使用 Apple 凭证登录。
3. 在左侧导航窗格中，选择 “证书 IDs 和配置文件”。
4. 在左侧导航窗格中，选择Identifiers（标识符）。
5. 在 Identifiers（标识符）页面上，选择 + 图标。
6. 在 “注册新标识符” 页面上，选择 “应用程序” IDs，然后选择 “继续”。
7. 在 Register an App ID（注册应用程序 ID）页面上，执行以下操作：

   a. 在 Description（描述）下方，键入描述。

b. 在 Bundle ID（服务包 ID）下，键入标识符。记下此 Bundle ID（捆绑包 ID），因为您需要此值才能将 Apple 配置为身份池中的提供商。

c. 在 Capabilities（功能）下，选择 Sign In with Apple，然后选择 Edit（编辑）。

d. 在通过 Apple 登录：应用程序 ID 配置页面上，为应用程序选择适当的设置。然后选择 Save（保存）。

e. 选择 Continue（继续）。

8. 在 Confirm your App ID（确认您的应用程序 ID）页面上，选择 Register（注册）。

9. 如果要将 Sign in with Apple 与本机 iOS 应用程序集成，请继续执行步骤 10。步骤 11 适用于您希望与 Sign in with Apple JS 集成的应用程序。

10. 在标识符页面上，选择应用程序 IDs菜单，然后选择服务 IDs。选择 + 图标。

11. 在 “注册新标识符” 页上，选择 “服务” IDs，然后选择 “继续”。

12. 在 Register an Services ID（注册服务 ID）页面上，执行以下操作：

    a. 在 Description（描述）下方，键入描述。

    b. 在 Identifier（标识符）下方，键入标识符。记下服务 ID，因为您需要此值才能将 Apple 配置为身份池中的提供商。

    c. 选择 Sign In with Apple（使用苹果账号登录），然后选择 Configure（配置）。

    d. 在 Web Authentication Configuration（Web 身份验证配置）页面上，选择 Primary App ID（主应用程序 ID）。在 “网站” 下 URLs，选择 + 图标。对于 Domains and Subdomains（域和子域），输入应用程序的域名。在 R eturn 中 URLs，输入回传网址，在用户通过 “使用 Apple 登录” 进行身份验证后，授权会将用户重定向到该网址。

    e. 选择下一步。

    f. 选择 Continue（继续），然后选择 Register（注册）。

13. 在左侧导航窗格中，选择 Keys（密钥）。

14. 在 Keys（密钥）页面上，选择 + 图标。

15. 在 Register a New Key（注册新密钥）页面上，执行以下操作：

    a. 在 Key Name（密钥名称）下方，键入密钥名称。

    b. 选择 Sign In with Apple，然后选择 Configure（配置）。

    c. 在 Configure Key（配置密钥）页面上，选择 Primary App ID（主应用程序 ID），然后选择 Save（保存）。

    d. 选择 Continue（继续），然后选择 Register（注册）。

> **Note**
>
> 要将 Sign in with Apple 与本机 iOS 应用程序集成，请参阅通过 Sign in with Apple 实施用户身份验证。
> 要在本机 iOS 以外的平台中集成 Sign in with Apple（使用苹果账号登录），请参阅使用 Apple JS 登录。

在 Amazon Cognito 联合身份控制台中配置外部提供商

使用以下过程可以配置外部提供商。

添加通过 Apple 登录身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择通过 Apple 登录。
5. 输入你使用 Apple 开发者创建的 OAuth 项目的服务 ID。有关更多信息，请参阅通过 Apple 登录文档中的使用通过 Apple 登录对用户进行身份验证。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择保存更改。

在 Amazon Cognito 联合身份 CLI 中以 Sign in with Apple 作为提供商的示例

此示例创建一个名为 `MyIdentityPool` 的身份池，并使用 Sign in with Apple（使用苹果账号登录）作为 IdP。

```
aws cognito-identity create-identity-pool --identity-
pool-name MyIdentityPool --supported-login-providers
appleid.apple.com="sameple.apple.clientid"
```

有关更多信息，请参阅创建身份池

生成 Amazon Cognito 身份 ID

此示例生成（或检索）Amazon Cognito ID。这是一个公有 API，因此您不需要任何凭证即可调用此 API。

```
aws cognito-identity get-id --identity-pool-id SampleIdentityPoolId --
logins appleid.apple.com="SignInWithAppleIdToken"
```

有关更多信息，请参阅 get-id。

获取 Amazon Cognito 身份 ID 的凭证

此示例返回用于提供的身份 ID 和 Sign in with Apple 登录的凭证。这是一个公有 API，因此您不需要任何凭证即可调用此 API。

```
aws cognito-identity get-credentials-for-identity --identity-id
SampleIdentityId --logins appleid.apple.com="SignInWithAppleIdToken"
```

有关更多信息，请参阅 get-credentials-for-identity。

使用 Sign in with Apple：Android

Apple 不提供支持 Sign in with Apple for Android 的开发工具包。您可以改为在 Web 视图中使用 Web 流。

- 要在应用程序中配置 Sign in with Apple，请按照 Apple 文档中的配置您的 Web 页面以使用 Sign in with Apple 操作。
- 要在你的安卓用户界面中添加 “用 Apple 登录” 按钮，请按照 Apple 文档中的在网页上显示 “使用 Apple 登录” 按钮进行操作。
- 要使用 Sign in with Apple（使用苹果账号登录）安全地对用户进行身份验证，请按照 Apple 文档中的使用 Sign in with Apple（使用苹果账号登录）对用户进行身份验证操作。

Sign in with Apple 使用会话对象跟踪其状态。Amazon Cognito 使用此会话对象中的 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

```
@Override
public void onSuccess(Bundle response) {
    String token = response.getString("id_token");
    Map<String, String> logins = new HashMap<String, String>();
    logins.put("appleid.apple.com", token);
    credentialsProvider.setLogins(logins);
}
```

使用 Sign in with Apple：iOS - Objective-C

Apple 为原生 iOS 应用程序中的 Sign in with Apple 提供了开发工具包支持。要在本机 iOS 设备中使用 Sign in with Apple 实施用户身份验证，请按照 Apple 文档中的使用 Sign in with Apple 实施用户身份验证操作。

Amazon Cognito 使用 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

```
(void)finishedWithAuth: (ASAuthorizationAppleIDCredential *)auth error: (NSError *)
 error {
        NSString *idToken = [ASAuthorizationAppleIDCredential
 objectForKey:@"identityToken"];
        credentialsProvider.logins = @{ "appleid.apple.com": idToken };
    }
```

所用 Sign in with Apple：iOS - Swift

Apple 为原生 iOS 应用程序中的 Sign in with Apple 提供了开发工具包支持。要在本机 iOS 设备中使用 Sign in with Apple 实施用户身份验证，请按照 Apple 文档中的使用 Sign in with Apple 实施用户身份验证操作。

Amazon Cognito 使用 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

有关如何在 iOS 中设置 Sign in with Apple（使用苹果账号登录）的更多信息，请参阅设置 Sign in with Apple（使用苹果账号登录）

```
func finishedWithAuth(auth: ASAuthorizationAppleIDCredential!, error: NSError!) {
    if error != nil {
      print(error.localizedDescription)
    }
    else {
      let idToken = auth.identityToken,
      credentialsProvider.logins = ["appleid.apple.com": idToken!]
    }
}
```

使用 “通过 Apple 登录”： JavaScript

Apple 不提供支持 “用苹果登录” 功能的 SDK JavaScript。您可以改为在 Web 视图中使用 Web 流。

- 要在应用程序中配置 Sign in with Apple，请按照 Apple 文档中的配置您的 Web 页面以使用 Sign in with Apple 操作。
- 要在 JavaScript 用户界面中添加 “使用 Apple 登录” 按钮，请按照 Apple 文档中的在网页上显示 “使用 Apple 登录” 按钮进行操作。
- 要使用 Sign in with Apple（使用苹果账号登录）安全地对用户进行身份验证，请按照 Apple 文档中的使用 Sign in with Apple（使用苹果账号登录）对用户进行身份验证操作。

Sign in with Apple 使用会话对象跟踪其状态。Amazon Cognito 使用此会话对象中的 ID 令牌对用户进行身份验证，生成唯一标识符，并在需要时授予用户访问其他 Amazon 资源的权限。

```
function signinCallback(authResult) {
     // Add the apple's id token to the Amazon Cognito credentials login map.
     AWS.config.credentials = new AWS.CognitoIdentityCredentials({
        IdentityPoolId: 'IDENTITY_POOL_ID',
        Logins: {
           'appleid.apple.com': authResult['id_token']
        }
     });

     // Obtain AWS credentials
```

```
    AWS.config.credentials.get(function(){
       // Access AWS resources here.
    });
}
```

设置 OIDC 提供者作为身份池 IdP

OpenID Connect 是许多登录提供程序支持的身份验证开放标准。借助 Amazon Cognito，您可以将身份与您通过 Amazon Identity and Access Management 配置的 OpenID Connect 提供者关联在一起。

添加 OpenID Connect 提供商

有关如何创建 OpenID Connect 提供者的信息，请参阅《Amazon Identity and Access Management 用户指南》中的创建 OpenID Connect（OIDC）身份提供者。

将提供商与 Amazon Cognito 关联

添加 OIDC 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 OpenID Connect（OIDC）。
5. 从您的 IAM 中选择一个 OIDC 身份提供商 IdPs 。 Amazon Web Services 账户如果您想添加新的 SAML 提供者，请选择创建新的提供者以导航到 IAM 控制台。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。

a. 如果不应用主体标签，请选择非活动。

b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。

c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。

8. 选择保存更改。

您可以将多个 OpenID Connect 提供商与一个身份池关联。

使用 OpenID Connect

有关如何登录和接收 ID 令牌的信息，请参阅提供商的文档。

拥有令牌后，将此令牌添加到登录映射。使用提供程序的 URI 作为密钥。

验证 OpenID Connect 令牌

首次与 Amazon Cognito 集成时，您可能会收到 `InvalidToken` 异常。务必要了解 Amazon Cognito 如何验证 OpenID Connect (OIDC) 令牌。

> **Note**
>
> 如此处所述 (https://tools.ietf.org/html/rfc7523)，Amazon Cognito 提供 5 分钟的宽限期来处理系统之间的任何时钟偏差。

1. `iss` 参数必须与登录映射使用的密钥匹配（如 login.provider.com）。
2. 签名必须有效。签名必须可通过 RSA 公有密钥进行验证。

> **Note**
>
> 身份池会将 OIDC IdP 签名密钥的缓存短时间保存。如果您的提供商更改了签名密钥，Amazon Cognito 可能会返回`NoKeyFound`错误，直到此缓存刷新为止。如果您遇到此错误，请等待大约十分钟，让您的身份池刷新签名密钥。

3. 证书公钥的指纹与您在创建 OIDC 提供程序时在 IAM 中设置的指纹相匹配。
4. 如果存在该`azp`参数，请根据您的 OIDC 提供商 IDs 中列出的客户端检查此值。

5. 如果azp参数不存在，请根据您的 OIDC 提供商 IDs 中列出的客户端检查该aud参数。

jwt.io 网站是用于解码令牌以验证这些值的宝贵资源。

Android

```
Map<String, String> logins = new HashMap<String, String>();
logins.put("login.provider.com", token);
credentialsProvider.setLogins(logins);
```

iOS - Objective-C

```
credentialsProvider.logins = @{ "login.provider.com": token }
```

JavaScript

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
 IdentityPoolId: 'IDENTITY_POOL_ID',
 Logins: {
    'login.provider.com': token
 }
});
```

设置 SAML 提供者作为身份池 IdP

借助 Amazon Cognito 身份池，您可以通过 SAML 2.0 使用身份提供商 (IdPs) 对用户进行身份验证。您可以使用支持 SAML 和 Amazon Cognito 的 IdP，为用户提供简单的引导流程。您支持 SAML 的 IdP 指定了用户可以承担的 IAM 角色。这样，不同的用户可以获得不同的权限集。

配置 SAML IdP 身份池

以下步骤介绍了如何配置身份池以使用基于 SAML 的 IdP。

> **Note**
>
> 在配置身份池以支持 SAML 提供商前，您必须先在 IAM 控制台中配置 SAML IdP。有关更多信息，请参阅 IAM 用户指南中的将第三方 SAML 解决方案提供商与 Amazon集成。

添加 SAML 身份提供者（IdP）

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择 SAML。
5. 从您 Amazon Web Services 账户的 IAM 中选择一个 SAML 身份提供商 IdPs 。如果您想添加新的 SAML 提供者，请选择创建新的提供者以导航到 IAM 控制台。
6. 要设置 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时请求的角色，请配置角色设置。
   - 您可以为该 IdP 中的用户分配您在配置经过身份验证的角色时设置的原定设置角色，也可以使用规则选择角色。
     i. 如果您选择使用规则选择角色，请输入用户身份验证中的来源声明、您要用来比较声明的运算符、导致与该角色选择匹配的值，以及当角色分配匹配时要分配的角色。选择添加其他，以根据不同的条件创建其他规则。
     ii. 选择角色解析。当用户的声明与您的规则不匹配时，您可以拒绝凭证或为经过身份验证的角色颁发凭证。
7. 要更改 Amazon Cognito 在向通过该提供者进行身份验证的用户颁发凭证时分配的主体标签，请配置访问控制属性。
   a. 如果不应用主体标签，请选择非活动。
   b. 要基于 sub 和 aud 声明应用主体标签，请选择使用原定设置映射。
   c. 要为主体标签创建自己的自定义属性模式，请选择使用自定义映射。然后，对于您要在标签中表示的每个声明，输入要从该声明中获取的标签键。
8. 选择保存更改。

配置 SAML IdP

创建 SAML 提供程序后，配置 SAML IdP，以在 IdP 和 Amazon之间添加信赖方信任。使用 man IdPs y，您可以指定一个 URL，IdP 可以使用该网址从 XML 文档中读取信赖方信息和证书。对于 Amazon，你可以使用 https://signin.aws.amazon.com/static/saml-metadata.xml。下一步是配置来自 IdP 的 SAML 断言响应，以填充所需的声明。 Amazon 有关申请配置的详细信息，请参阅针对身份验证响应配置 SAML 断言。

如果您的 SAML IdP 在 SAML 元数据中包含多个签名证书，则在登录时，如果与 SAML 元数据中的任何证书匹配，您的身份池就会确定 SAML 断言有效。

使用 SAML 自定义用户角色

将 SAML 与 Amazon Cognito 身份结合使用时，可针对终端用户自定义角色。Amazon Cognito 只支持对基于 SAML 的 IdP 使用增强流程。您无需为身份池指定经过身份验证或未经身份验证的角色，即可使用基于 SAML 的 IdP。`https://aws.amazon.com/SAML/Attributes/Role` 声明属性指定一个或多个逗号分隔的角色和提供商 ARN 对。这些是用户可以担任的角色。您可以配置 SAML IdP 以根据 IdP 提供的用户属性信息填充角色属性。如果您在 SAML 断言中收到多个角色，请在调用 `getCredentialsForIdentity` 时填充可选的 `customRoleArn` 参数。如果 `customRoleArn` 角色与 SAML 断言中的声明中的角色匹配，则用户将承担此角色。

使用 SAML IdP 对用户进行身份验证

要与基于 SAML 的 IdP 联合，请确定用户启动登录的 URL。 Amazon 联盟使用 IDP 发起的登录。在 AD FS 2.0 中，URL 采用 `https://<fqdn>/adfs/ls/IdpInitiatedSignOn.aspx?loginToRp=urn:amazon:webservices` 格式。

要在 Amazon Cognito 中添加对 SAML IdP 的支持，请首先使用 SAML 身份提供商从 iOS 或 Android 应用程序对用户进行身份验证。您用于与 SAML IdP 集成和向其进行身份验证的代码特定于 SAML 提供商。在对用户进行身份验证后，您可以使用 Amazon Cognito APIs 向亚马逊 Cognito Identity 提供生成的 SAML 断言。

您无法在身份池 API 请求的 `Logins` 映射中重复或重放 SAML 断言。重放的 SAML 断言的断言 ID 与早期 API 请求的 ID 重复。可以在`Logins`地图中接受 SAML 断言的 API 操作包括GetId、GetCredentialsForIdentityGetOpenIdToken、和。GetOpenIDTokenForDeveloperIdentity您可以在身份池身份验证流程中对于每个 API 请求重放 SAML 断言 ID 一次。例如，您可以在 `GetId` 请求和后续 `GetCredentialsForIdentity` 请求中提供相同的 SAML 断言，但不能在第二个 `GetId` 请求中提供相同的 SAML 断言。

经开发人员验证的身份

除了通过 将 Facebook 设置为身份池 IdP、将 Google 设置为身份池 IdP、设置 Login with Amazon 作为身份池 IdP 和 使用 Apple 作为身份池 IdP 来设置登录 的 Web 身份联合验证之外，Amazon Cognito 还支持经开发人员验证的身份。使用经过开发人员身份验证的身份，您可以通过自己的现有身份验证流程注册和验证用户，同时仍可以使用 Amazon Cognito 同步用户数据和访问资源。 Amazon 使用经开发人员验证的身份涉及最终用户设备、身份验证后端和 Amazon Cognito 之间的交互。有关更多详细信

息，请参阅博客中的了解 Amazon Cognito 身份验证第 2 部分：经过开发人员身份验证的 Amazon 身份。

了解身份验证流程

GetOpenIdTokenForDeveloperIdentityAPI 操作可以为增强身份验证和基本身份验证启动开发者身份验证。此 API 使用管理凭证对请求进行身份验证。`Logins` 映射是身份池开发人员提供者的名称，例如与自定义标识符配对的 `login.mydevprovider`。

示例：

```
"Logins": {
        "login.mydevprovider": "my developer identifier"
    }
```

增强型身份验证

使用包含令牌名称`cognito-identity.amazonaws.com`和值`Logins`的地图调用 GetCredentialsForIdentityAPI 操作`GetOpenIdTokenForDeveloperIdentity`。

示例：

```
"Logins": {
        "cognito-identity.amazonaws.com": "eyJra12345EXAMPLE"
    }
```

`GetCredentialsForIdentity` 使用经开发人员验证的身份，它会返回身份池中默认经过身份验证的角色的临时凭证。

基本身份验证

调用 AssumeRoleWithWebIdentityAPI 操作并请求已定义适当信任关系`RoleArn`的任何 IAM 角色的。将 `WebIdentityToken` 的值设置为从 `GetOpenIdTokenForDeveloperIdentity` 获取令牌。

要了解经开发人员验证的身份的身份验证流程以及该流程与外部提供者身份验证流程有何不同，请参阅身份池身份验证流程。

定义开发人员提供商名称并将其与身份池关联

要使用经开发人员验证的身份，您需要与开发人员提供者关联的身份池。为此，请按照以下步骤操作：

添加自定义开发人员提供者

1. 从 Amazon Cognito 控制台中选择身份池。选择身份池。
2. 选择用户访问选项卡。
3. 选择添加身份提供者。
4. 选择自定义开发人员提供者。
5. 输入开发人员提供者名称。添加开发人员提供者后，无法更改或删除它。
6. 选择保存更改。

注意：一旦设置提供商名称，便无法进行更改。

实施身份提供商

Android

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 `AWSAbstractCognitoIdentityProvider`。您的身份提供商类应返回包含令牌作为属性的响应对象。

以下是身份提供者的基本示例。

```
public class DeveloperAuthenticationProvider extends
 AWSAbstractCognitoDeveloperIdentityProvider {

  private static final String developerProvider = "<Developer_provider_name>";

  public DeveloperAuthenticationProvider(String accountId, String identityPoolId,
 Regions region) {
    super(accountId, identityPoolId, region);
    // Initialize any other objects needed here.
  }

  // Return the developer provider name which you choose while setting up the
  // identity pool in the &COG; Console

  @Override
  public String getProviderName() {
    return developerProvider;
  }
```

```
  // Use the refresh method to communicate with your backend to get an
  // identityId and token.

  @Override
  public String refresh() {

    // Override the existing token
    setToken(null);

    // Get the identityId and token by making a call to your backend
    // (Call to your backend)

    // Call the update method with updated identityId and token to make sure
    // these are ready to be used from Credentials Provider.

    update(identityId, token);
    return token;

  }

  // If the app has a valid identityId return it, otherwise get a valid
  // identityId from your backend.

  @Override
  public String getIdentityId() {

    // Load the identityId from the cache
    identityId = cachedIdentityId;

    if (identityId == null) {
       // Call to your backend
    } else {
       return identityId;
    }

  }
}
```

要使用此身份提供商，您必须将其传递到 `CognitoCachingCredentialsProvider`。示例如下：

```
DeveloperAuthenticationProvider developerProvider = new
 DeveloperAuthenticationProvider( null, "IDENTITYPOOLID", context, Regions.USEAST1);
```

```
CognitoCachingCredentialsProvider credentialsProvider = new
 CognitoCachingCredentialsProvider( context, developerProvider, Regions.USEAST1);
```

iOS – objective-C

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 AWSCognitoCredentialsProviderHelper。您的身份提供商类应返回包含令牌作为属性的响应对象。

```
@implementation DeveloperAuthenticatedIdentityProvider
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */

- (AWSTask <NSString*> *) token {
    //Write code to call your backend:
    //Pass username/password to backend or some sort of token to authenticate user
    //If successful, from backend call getOpenIdTokenForDeveloperIdentity with logins
 map
    //containing "your.provider.name":"enduser.username"
    //Return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = response.identityId;
    return [AWSTask taskWithResult:response.token];
}

@end
```

要使用此身份提供商，请将其传递到 `AWSCognitoCredentialsProvider`，如下例所示：

```
DeveloperAuthenticatedIdentityProvider * devAuth =
 [[DeveloperAuthenticatedIdentityProvider alloc]
 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
                                          identityPoolId:@"YOUR_IDENTITY_POOL_ID"
                                         useEnhancedFlow:YES
                                identityProviderManager:nil];
AWSCognitoCredentialsProvider *credentialsProvider = [[AWSCognitoCredentialsProvider
 alloc]

 initWithRegionType:AWSRegionYOUR_IDENTITY_POOL_REGION
```

```
                                                                identityProvider:devAuth];
```

如果您想同时支持未经身份验证的身份和经开发人员验证的身份，请在 `logins` 实施中覆盖 `AWSCognitoCredentialsProviderHelper` 方法。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else{
        return [super logins];
    }
}
```

如果您想支持经开发人员验证的身份和社交提供者，您必须管理在 `AWSCognitoCredentialsProviderHelper` 的 `logins` 实施中谁是当前的提供者。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

iOS – swift

要使用经开发人员验证的身份，请实施自己的身份提供者类，该类可扩展 AWSCognitoCredentialsProviderHelper。您的身份提供商类应返回包含令牌作为属性的响应对象。

```
import AWSCore
/*
 * Use the token method to communicate with your backend to get an
 * identityId and token.
 */
class DeveloperAuthenticatedIdentityProvider : AWSCognitoCredentialsProviderHelper {
    override func token() -> AWSTask<NSString> {
    //Write code to call your backend:
    //pass username/password to backend or some sort of token to authenticate user, if
 successful,
```

```
    //from backend call getOpenIdTokenForDeveloperIdentity with logins map containing
 "your.provider.name":"enduser.username"
    //return the identity id and token to client
    //You can use AWSTaskCompletionSource to do this asynchronously

    // Set the identity id and return the token
    self.identityId = resultFromAbove.identityId
    return AWSTask(result: resultFromAbove.token)
}
```

要使用此身份提供商，请将其传递到 `AWSCognitoCredentialsProvider`，如下例所示：

```
let devAuth =
 DeveloperAuthenticatedIdentityProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityPoolId: "YOUR_IDENTITY_POOL_ID", useEnhancedFlow: true,
 identityProviderManager:nil)
let credentialsProvider =
 AWSCognitoCredentialsProvider(regionType: .YOUR_IDENTITY_POOL_REGION,
 identityProvider:devAuth)
let configuration = AWSServiceConfiguration(region: .YOUR_IDENTITY_POOL_REGION,
 credentialsProvider:credentialsProvider)
AWSServiceManager.default().defaultServiceConfiguration = configuration
```

如果您想同时支持未经身份验证的身份和经开发人员验证的身份，请在 `logins` 实施中覆盖 `AWSCognitoCredentialsProviderHelper` 方法。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else {
        return super.logins()
    }
}
```

如果您想支持经开发人员验证的身份和社交提供者，您必须管理在 `AWSCognitoCredentialsProviderHelper` 的 `logins` 实施中谁是当前的提供者。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
```

```
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

JavaScript

从后端获取身份 ID 和会话令牌后，您要将它们传递到 `AWS.CognitoIdentityCredentials` 提供者。以下为示例。

```
AWS.config.credentials = new AWS.CognitoIdentityCredentials({
   IdentityPoolId: 'IDENTITY_POOL_ID',
   IdentityId: 'IDENTITY_ID_RETURNED_FROM_YOUR_PROVIDER',
   Logins: {
      'cognito-identity.amazonaws.com': 'TOKEN_RETURNED_FROM_YOUR_PROVIDER'
   }
});
```

Unity

要使用经开发人员验证的身份，您必须扩展 `CognitoAWSCredentials` 并覆盖 `RefreshIdentity` 方法，以从后端检索用户身份 ID 和令牌，并将它们返回。下面是可通过“example.com”联系假想后端的身份提供者的简单示例：

```
using UnityEngine;
using System.Collections;
using Amazon.CognitoIdentity;
using System.Collections.Generic;
using ThirdParty.Json.LitJson;
using System;
using System.Threading;

public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
```

```
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }

    protected override IdentityState RefreshIdentity()
    {
        IdentityState state = null;
        ManualResetEvent waitLock = new ManualResetEvent(false);
        MainThreadDispatcher.ExecuteCoroutineOnMainThread(ContactProvider((s) =>
        {
            state = s;
            waitLock.Set();
        }));
        waitLock.WaitOne();
        return state;
    }

    IEnumerator ContactProvider(Action<IdentityState> callback)
    {
        WWW www = new WWW("http://example.com/?username="+login);
        yield return www;
        string response = www.text;

        JsonData json = JsonMapper.ToObject(response);

        //The backend has to send us back an Identity and a OpenID token
        string identityId = json["IdentityId"].ToString();
        string token = json["Token"].ToString();

        IdentityState state = new IdentityState(identityId, PROVIDER_NAME, token,
 false);
        callback(state);
    }
}
```

上面的代码使用线程调度程序对象调用协同程序。如果您在项目中无法执行上述操作，您可以在场景中使用以下脚本：

```
using System;
```

```
using UnityEngine;
using System.Collections;
using System.Collections.Generic;

public class MainThreadDispatcher : MonoBehaviour
{
    static Queue<IEnumerator> _coroutineQueue = new Queue<IEnumerator>();
    static object _lock = new object();

    public void Update()
    {
        while (_coroutineQueue.Count > 0)
        {
            StartCoroutine(_coroutineQueue.Dequeue());
        }
    }

    public static void ExecuteCoroutineOnMainThread(IEnumerator coroutine)
    {
        lock (_lock) {
            _coroutineQueue.Enqueue(coroutine);
        }
    }
}
```

Xamarin

要使用经开发人员验证的身份，您必须扩展 `CognitoAWSCredentials` 并覆盖 `RefreshIdentity` 方法，以从后端检索用户身份 ID 和令牌，并将它们返回。下面是可通过“example.com”联系假想后端的身份提供者的基本示例：

```
public class DeveloperAuthenticatedCredentials : CognitoAWSCredentials
{
    const string PROVIDER_NAME = "example.com";
    const string IDENTITY_POOL = "IDENTITY_POOL_ID";
    static readonly RegionEndpoint REGION = RegionEndpoint.USEast1;
    private string login = null;

    public DeveloperAuthenticatedCredentials(string loginAlias)
        : base(IDENTITY_POOL, REGION)
    {
        login = loginAlias;
    }
```

```
    protected override async Task<IdentityState> RefreshIdentityAsync()
    {
        IdentityState state = null;
        //get your identity and set the state
        return state;
    }
}
```

更新登录映射（仅限 Android 和 iOS）

Android

使用身份验证系统成功对用户进行身份验证后，请使用开发人员提供者名称和开发人员用户标识符更新登录映射。此标识符是一个字母数字字符串，可在身份验证系统中唯一标识用户。请确保在更新登录映射后调用 `refresh` 方法，因为 `identityId` 可能已更改：

```
HashMap<String, String> loginsMap = new HashMap<String, String>();
loginsMap.put(developerAuthenticationProvider.getProviderName(),
 developerUserIdentifier);

credentialsProvider.setLogins(loginsMap);
credentialsProvider.refresh();
```

iOS – objective-C

如果没有凭证或者凭证已过期，则 iOS 开发工具包仅调用 `logins` 方法，以获取最新登录映射。如果您要强制 SDK 获取新的凭证（例如，最终用户从未经身份验证变为经过身份验证并且您想要经过身份验证的用户的凭证），则对 `credentialsProvider` 调用 `clearCredentials`。

```
[credentialsProvider clearCredentials];
```

iOS – swift

如果没有凭证或者凭证已过期，则 iOS 开发工具包仅调用 `logins` 方法，以获取最新登录映射。如果您要强制开发工具包获取新的凭证 (例如，最终用户从未经身份验证变为经过身份验证并且您想要经过身份验证的用户的凭证)，则在 `clearCredentials` 上调用 `credentialsProvider`。

```
credentialsProvider.clearCredentials()
```

获取令牌（服务器端）

您可以通过调用获取令牌GetOpenIdTokenForDeveloperIdentity。必须使用 Amazon 开发者凭据从您的后端调用此 API。不得从客户端开发工具包调用它。API 接收 Cognito 身份池 ID；包含身份提供者名称作为密钥及标识符作为值的登录映射；以及可选 Cognito 身份 ID（例如，您让一个未经过身份验证的用户变成了经身份验证的用户）。标识符可以是用户的用户名、电子邮件地址或数值。API 通过为用户提供唯一 Cognito ID 及为最终用户提供 OpenID Connect 令牌来响应您的调用。

对于由 `GetOpenIdTokenForDeveloperIdentity` 返回的令牌，您需要注意以下事项：

- 您可以指定令牌的自定义过期时间，以便缓存。如果您不提供任何自定义过期时间，则令牌的有效期为 15 分钟。
- 您可以设置的最大令牌持续时间为 24 小时。
- 请留意延长令牌持续时间所带来的安全方面的问题。如果攻击者获得此令牌，他们可以在令牌有效期内将其交换为最终用户的 Amazon 凭证。

以下 Java 代码段显示了如何初始化 Amazon Cognito 客户端，以及如何检索经开发人员验证的身份的令牌。

```
// authenticate your end user as appropriate
// ....

// if authenticated, initialize a cognito client with your AWS developer credentials
AmazonCognitoIdentity identityClient = new AmazonCognitoIdentityClient(
  new BasicAWSCredentials("access_key_id", "secret_access_key")
);

// create a new request to retrieve the token for your end user
GetOpenIdTokenForDeveloperIdentityRequest request =
  new GetOpenIdTokenForDeveloperIdentityRequest();
request.setIdentityPoolId("YOUR_COGNITO_IDENTITY_POOL_ID");

request.setIdentityId("YOUR_COGNITO_IDENTITY_ID"); //optional, set this if your client
 has an
                                                    //identity ID that you want to link
 to this
                                                    //developer account

// set up your logins map with the username of your end user
HashMap<String,String> logins = new HashMap<>();
```

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
request.setLogins(logins);

// optionally set token duration (in seconds)
request.setTokenDuration(60 * 15l);
GetOpenIdTokenForDeveloperIdentityResult response =
  identityClient.getOpenIdTokenForDeveloperIdentity(request);

// obtain identity id and token to return to your client
String identityId = response.getIdentityId();
String token = response.getToken();

//code to return identity id and token to client
//...
```

按照上述步骤操作，您应该能够将经开发人员验证的身份集成到应用程序中。如有任何问题或疑问，请随时在我们的论坛上发帖。

连接到现有社交身份

当您使用经开发人员验证的身份时，您必须从后端链接所有提供者。要将自定义身份与用户的社交身份（Login with Amazon、使用 Apple 登录、Facebook 或 Google 登录）关联起来，请在致电GetOpenIdTokenForDeveloperIdentity时将身份提供者令牌添加到登录地图中。要实现上述目标，当您从客户端开发工具包调用后端来对最终用户进行身份验证时，您还需要传递最终用户的社交提供商令牌。

例如，如果您想将自定义身份链接到 Facebook，在调用 `GetOpenIdTokenForDeveloperIdentity` 时，除了身份提供商标识符之外，您还需要将 Facebook 令牌添加到登录映射。

```
logins.put("YOUR_IDENTITY_PROVIDER_NAME","YOUR_END_USER_IDENTIFIER");
logins.put("graph.facebook.com","END_USERS_FACEBOOK_ACCESSTOKEN");
```

支持在提供商之间转换

Android

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而

不是与身份验证系统联系。因此，移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供者做出一些更改。

`refresh` 方法检查登录映射。如果映射不为空并且有带开发人员提供者名称的密钥，请调用您的后端。否则，调用该 getIdentityId方法并返回 null。

```
public String refresh() {

   setToken(null);

   // If the logins map is not empty make a call to your backend
   // to get the token and identityId
   if (getProviderName() != null &&
      !this.loginsMap.isEmpty() &&
      this.loginsMap.containsKey(getProviderName())) {

      /**
       * This is where you would call your backend
       **/

      // now set the returned identity id and token in the provider
      update(identityId, token);
      return token;

   } else {
      // Call getIdentityId method and return null
      this.getIdentityId();
      return null;
   }
}
```

同样，`getIdentityId` 方法也有两个流程，具体取决于登录映射的内容：

```
public String getIdentityId() {

   // Load the identityId from the cache
   identityId = cachedIdentityId;

   if (identityId == null) {

      // If the logins map is not empty make a call to your backend
      // to get the token and identityId
```

```
      if (getProviderName() != null && !this.loginsMap.isEmpty()
         && this.loginsMap.containsKey(getProviderName())) {

         /**
           * This is where you would call your backend
          **/

         // now set the returned identity id and token in the provider
         update(identityId, token);
         return token;

      } else {
         // Otherwise call &COG; using getIdentityId of super class
         return super.getIdentityId();
      }

   } else {
      return identityId;
   }

}
```

iOS – objective-C

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。为此，请重写该AWSCognitoCredentialsProviderHelper`logins`方法，以便能够根据当前身份提供者返回正确的登录映射。此示例说明如何能够在未经身份验证的身份、Facebook 和经开发人员验证的身份之间切换。

```
- (AWSTask<NSDictionary<NSString *, NSString *> *> *)logins {
    if(/*logic to determine if user is unauthenticated*/) {
        return [AWSTask taskWithResult:nil];
    }else if (/*logic to determine if user is Facebook*/){
        return [AWSTask taskWithResult: @{ AWSIdentityProviderFacebook :
 [FBSDKAccessToken currentAccessToken] }];
    }else {
        return [super logins];
    }
}
```

当您从未经身份验证转换为经过身份验证时，您应该调用 `[credentialsProvider clearCredentials];` 以强制开发工具包获取经过身份验证的新凭证。当您在两个经过身份验证的

提供者之间切换并且不想将这两个提供者链接起来时（例如，您没有在登录词典中为多个提供者提供令牌），请调用 `[credentialsProvider clearKeychain];`。上述操作将清除凭证和身份，并强制开发工具包获取新的。

iOS – swift

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。为此，请重写该AWSCognitoCredentialsProviderHelper`logins`方法，以便能够根据当前身份提供者返回正确的登录映射。此示例说明如何能够在未经身份验证的身份、Facebook 和经开发人员验证的身份之间切换。

```
override func logins () -> AWSTask<NSDictionary> {
    if(/*logic to determine if user is unauthenticated*/) {
        return AWSTask(result:nil)
    }else if (/*logic to determine if user is Facebook*/){
        if let token = AccessToken.current?.authenticationToken {
            return AWSTask(result: [AWSIdentityProviderFacebook:token])
        }
        return AWSTask(error:NSError(domain: "Facebook Login", code: -1 , userInfo:
 ["Facebook" : "No current Facebook access token"]))
    }else {
        return super.logins()
    }
}
```

当您从未经身份验证转换为经过身份验证时，您应该调用 `credentialsProvider.clearCredentials()` 以强制开发工具包获取经过身份验证的新凭证。当您在两个经过身份验证的提供商之间切换并且不想将这两个提供商链接起来时 (即，您没有在登录词典中为多个提供商提供令牌)，您应该调用 `credentialsProvider.clearKeychain()`。上述操作将清除凭证和身份，并强制开发工具包获取新的。

Unity

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而不是与身份验证系统联系。移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供商做出一些更改。

在 Unity 中执行此操作的推荐方法是从 AmazonCognitoEnhancedIdentityProvide 而不是扩展您的身份提供商 AbstractCognitoIdentityProvider，并调用父 RefreshAsync 方法而不是您自己的方法，以防用户未使用您自己的后端进行身份验证。如果用户已经过身份验证，则您可以使用之前介绍的相同的流程。

Xamarin

您的应用程序可能需要支持未经身份验证的身份或使用公有提供者（Login with Amazon、通过 Apple 登录、Facebook 或 Google）的经过身份验证的身份，以及经开发人员验证的身份。经开发人员验证的身份与其他身份（未经身份验证的身份和使用公共提供者的经过身份验证的身份）的主要区别在于 identityId 和令牌的获取方式。对于其他身份，移动应用程序将直接与 Amazon Cognito 进行交互，而不是与身份验证系统联系。移动应用程序应该能够支持两个不同的流程，具体取决于应用程序用户的选择。对此，您必须对自定义身份提供者做出一些更改。

将未经身份验证的用户切换为经过身份验证的用户

Amazon Cognito 身份池同时支持经过身份验证的用户和未经身份验证的用户。未经身份验证的用户即使未使用您的任何身份提供商登录，也可以访问您的 Amazon 资源（IdPs）。此级别的访问可用于向尚未登录的用户显示内容。即使每个未经身份验证的用户尚未单独登录和经过身份验证，这些用户在身份池中也都具有唯一的身份。

本节介绍了用户如何选择从使用未经身份验证的身份登录切换为使用经过身份验证的身份登录。

Android

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能会决定使用其中一个支持的登录 IdPs。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

应用程序会通过 `IdentityChangedListener` 界面收到配置文件合并的消息。在界面中实施 `identityChanged` 方法以接收这些消息：

```
@override
public void identityChanged(String oldIdentityId, String newIdentityId) {
    // handle the change
}
```

iOS – objective-C

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能会决定使用其中一个支持的登录 IdPs。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

`NSNotificationCenter` 通知应用程序配置文件合并的消息：

```
[[NSNotificationCenter defaultCenter] addObserver:self
                                      selector:@selector(identityIdDidChange:)
                                      name:AWSCognitoIdentityIdChangedNotification
                                      object:nil];

-(void)identityDidChange:(NSNotification*)notification {
    NSDictionary *userInfo = notification.userInfo;
    NSLog(@"identity changed from %@ to %@",
        [userInfo objectForKey:AWSCognitoNotificationPreviousId],
        [userInfo objectForKey:AWSCognitoNotificationNewId]);
}
```

iOS – swift

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能会决定使用其中一个支持的登录 IdPs。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

`NSNotificationCenter` 通知应用程序配置文件合并的消息：

```
[NSNotificationCenter.defaultCenter().addObserver(observer: self
   selector:"identityDidChange"
   name:AWSCognitoIdentityIdChangedNotification
   object:nil)

func identityDidChange(notification: NSNotification!) {
  if let userInfo = notification.userInfo as? [String: AnyObject] {
    print("identity changed from: \(userInfo[AWSCognitoNotificationPreviousId])
    to: \(userInfo[AWSCognitoNotificationNewId])")
  }
}
```

JavaScript

最初未经身份验证的用户

用户最初通常具有未经身份验证的角色。对于此角色，您可以设置配置对象的凭证属性，而不设置登录属性。在这种情况下，您的默认配置可能如下所示：

```
// set the default config object
var creds = new AWS.CognitoIdentityCredentials({
    IdentityPoolId: 'us-east-1:1699ebc0-7900-4099-b910-2df94f52a030'
});
AWS.config.credentials = creds;
```

切换为经过身份验证的用户

当未经身份验证的用户登录 IdP 并且您拥有令牌时，您可以通过调用可更新凭证对象和添加登录令牌的自定义函数，来将用户从未经身份验证的用户切换为经过身份验证的用户：

```
// Called when an identity provider has a token for a logged in user
function userLoggedIn(providerName, token) {
    creds.params.Logins = creds.params.Logins || {};
    creds.params.Logins[providerName] = token;

    // Expire credentials to refresh them on the next request
    creds.expired = true;
}
```

您还可以创建 `CognitoIdentityCredentials` 对象。在这种情况下，必须重置任何现有服务对象的凭证属性，以反映更新的凭证配置信息。请参阅使用全局配置对象。

有关该`CognitoIdentityCredentials`对象的更多信息，请参阅Amazon。CognitoIdentityCredentials在 适用于 JavaScript 的 Amazon SDK API 参考中。

Unity

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能会决定使用其中一个支持的登录 IdPs。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

您可以订阅 `IdentityChangedEvent`，以接收配置文件合并的通知：

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e)
{
    // handle the change
    Debug.log("Identity changed from " + e.OldIdentityId + " to " + e.NewIdentityId);
};
```

Xamarin

用户能够以未经身份验证的来宾的身份登录您的应用程序。最终，他们可能会决定使用其中一个支持的登录 IdPs。Amazon Cognito 将确保旧身份保留与新身份相同的唯一标识符，并确保配置文件数据自动合并。

```
credentialsProvider.IdentityChangedEvent += delegate(object sender,
 CognitoAWSCredentials.IdentityChangedArgs e){
    // handle the change
    Console.WriteLine("Identity changed from " + e.OldIdentityId + " to " +
 e.NewIdentityId);
};
```

Amazon Cognito Sync

> 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Sync 是一个 Amazon Web Services 服务 和客户端库，它使跨设备同步与应用程序相关的用户数据成为可能。Amazon Cognito 可以跨移动设备和 Web 同步用户配置文件数据，无需使用您自己的后端。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据。如果您设置推送同步，您可在更新可用时立即通知其他设备。

有关 Amazon Cognito 身份区域可用性的信息，请参阅Amazon 服务区域可用性。

要了解有关 Amazon Cognito Sync 的更多信息，请参阅以下主题。

主题

- Amazon Cognito Sync 入门
- 同步不同客户端的数据
- 处理事件回调
- 实施推送同步
- 实施 Amazon Cognito Sync 流
- 使用 Amazon Cognito Events 自定义工作流

Amazon Cognito Sync 入门

> 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Sync 是一项 Amazon 服务和客户端库，可实现与应用程序相关的用户数据的跨设备同步。您可以使用它来跨移动设备和 Web 应用程序同步用户配置文件数据。客户端库在本地缓存数据，因此，您的应用程序可以读取和写入数据，无论设备是否处于连接状态，都是如此。设备处于在线状态时，您可以同步数据，如果您设置推送同步，则在更新可用时会立即通知其他设备。

设置 Amazon Cognito 中的身份池

Amazon Cognito Sync 需要一个 Amazon Cognito 身份池来提供用户身份。您需要先设置身份池，然后才能使用 Amazon Cognito Sync。要创建身份池并安装开发工具包，请参阅Amazon Cognito 身份池入门。

存储和同步数据

设置身份池并安装开发工具包后，您就可以开始存储并在设备之间同步数据了。有关更多信息，请参阅同步不同客户端的数据。

同步不同客户端的数据

> 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

借助 Amazon Cognito，您可将用户数据保存在包含键/值对的数据集中。Amazon Cognito 将此数据与您身份池中的身份相关联，这样您的应用程序就可以跨登录和设备访问它。要在 Amazon Cognito 服务和终端用户设备之间同步此数据，请调用 synchronize 方法。每个数据集的最大大小为 1MB。您最多可以将 20 个数据集与一个身份关联。

Amazon Cognito Sync 客户端会为身份数据创建一个本地缓存。您的应用程序在读取和写入键时，它与此本地缓存通信。此通信保证您在设备上所做的所有更改都能在设备上立即可用，即使您处于离线状态也是如此。调用 synchronize 方法后，服务的更改将拉取到设备上，并且所有本地更改都会推送到该服务。此时，这些更改可供其他设备同步。

初始化 Amazon Cognito Sync 客户端

要初始化 Amazon Cognito Sync 客户端，您必须先创建凭证提供程序。凭证提供者会获取临时 Amazon 证书，以便您的应用程序可以访问您的 Amazon 资源。您还必须导入必要的标头文件。使用以下步骤初始化 Amazon Cognito Sync 客户端。

Android

1. 按照获取凭证中的说明创建凭证提供程序。
2. 如下所示导入 Amazon Cognito 程序包：`import com.amazonaws.mobileconnectors.cognito.*;`
3. 初始化 Amazon Cognito Sync。传入 Android 应用程序上下文、身份池 ID、 Amazon Web Services 区域和初始化的 Amazon Cognito 凭证提供程序，如下所示：

```
CognitoSyncManager client = new CognitoSyncManager(
    getApplicationContext(),
    Regions.YOUR_REGION,
    credentialsProvider);
```

iOS - Objective-C

1. 按照获取凭证中的说明创建凭证提供程序。
2. 导入 `AWSCore` 和 `Cognito`，并初始化 `AWSCognito`如下所示：

```
#import <AWSiOSSDKv2/AWSCore.h>
#import <AWSCognitoSync/Cognito.h>

AWSCognito *syncClient = [AWSCognito defaultCognito];
```

3. 如果您正在使用 CocoaPods，请`<AWSiOSSDKv2/AWSCore.h>`替换为`AWSCore.h`。请遵循与 Amazon Cognito 导入相同的语法。

iOS - Swift

1. 按照获取凭证中的说明创建凭证提供程序。
2. 导入并初始化 `AWSCognito`，如下所示：

```
import AWSCognito
let syncClient = AWSCognito.default()!
```

JavaScript

1. 下载适用于 Amazon Cognito 的同步管理器。 JavaScript
2. 在您的项目中添加 Sync Manager 库。
3. 按照获取凭证中的说明创建凭证提供程序。
4. 如下所示初始化 Sync Manager：

```
var syncManager = new AWS.CognitoSyncManager();
```

Unity

1. 按照获取凭证中的说明创建 `CognitoAWSCredentials` 的实例。
2. 创建 `CognitoSyncManager` 的实例。传递 `CognitoAwsCredentials` 对象和 `AmazonCognitoSyncConfig`，并至少包括区域集，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

Xamarin

1. 按照 获取凭证 中的说明创建 `CognitoAWSCredentials` 的实例。
2. 创建 `CognitoSyncManager` 的实例。传递 `CognitoAwsCredentials` 对象和 `AmazonCognitoSyncConfig`，并至少包括区域集，如下所示：

```
AmazonCognitoSyncConfig clientConfig = new AmazonCognitoSyncConfig { RegionEndpoint =
 REGION };
CognitoSyncManager syncManager = new CognitoSyncManager(credentials, clientConfig);
```

了解数据集

Amazon Cognito 将用户配置文件数据组织到数据集中。每个数据集可以包含高达 1MB 的键值对形式的数据。数据集是您可以同步的最细粒度实体。在调用 synchronize 方法之前，在数据集上执行的读取和写入操作只会对本地存储产生影响。Amazon Cognito 使用唯一字符串标识数据集。您可以创建新数据集或打开现有数据集，如下所示。

Android

```
Dataset dataset = client.openOrCreateDataset("datasetname");
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 `synchronize` 方法从 Amazon Cognito 中删除数据集，如下所示：

```
dataset.delete();
dataset.synchronize(syncCallback);
```

iOS - Objective-C

```
AWSCognitoDataset *dataset = [syncClient openOrCreateDataset:@"myDataSet"];
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 `synchronize` 方法从 Amazon Cognito 中删除数据集，如下所示：

```
[dataset clear];
[dataset synchronize];
```

iOS - Swift

```
let dataset = syncClient.openOrCreateDataset("myDataSet")!
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 `synchronize` 方法如下所示以从 Amazon Cognito 中删除数据集：

```
dataset.clear()
dataset.synchronize()
```

JavaScript

```
syncManager.openOrCreateDataset('myDatasetName', function(err, dataset) {
   // ...
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

要从数据集中删除键，请如下所示使用 `Remove`：

```
dataset.Remove("myKey");
```

Xamarin

```
Dataset dataset = syncManager.OpenOrCreateDataset("myDatasetName");
```

要删除数据集，首先要调用方法以将其从本地存储中删除，然后调用 `synchronize` 方法从 Amazon Cognito 中删除数据集，如下所示：

```
dataset.Delete();
dataset.SynchronizeAsync();
```

在数据集中读取并写入数据

Amazon Cognito 数据集的功能与字典一样，包含可以通过键访问的值。您可以读取、添加或修改数据集的键和值，就像数据集是字典一样，如下所示。

请注意，在您调用同步方法之前，写入数据集的值仅对本地缓存的数据副本有影响。

Android

```
String value = dataset.get("myKey");
dataset.put("myKey", "my value");
```

iOS - Objective-C

```
[dataset setString:@"my value" forKey:@"myKey"];
NSString *value = [dataset stringForKey:@"myKey"];
```

iOS - Swift

```
dataset.setString("my value", forKey:"myKey")
let value = dataset.stringForKey("myKey")
```

JavaScript

```
dataset.get('myKey', function(err, value) {
  console.log('myRecord: ' + value);
});

dataset.put('newKey', 'newValue', function(err, record) {
  console.log(record);
});

dataset.remove('oldKey', function(err, record) {
  console.log(success);
});
```

Unity

```
string myValue = dataset.Get("myKey");
dataset.Put("myKey", "newValue");
```

Xamarin

```
//obtain a value
string myValue = dataset.Get("myKey");

// Create a record in a dataset and synchronize with the server
dataset.OnSyncSuccess += SyncSuccessCallback;
dataset.Put("myKey", "myValue");
dataset.SynchronizeAsync();
```

```
void SyncSuccessCallback(object sender, SyncSuccessEventArgs e) {
  // Your handler code here
}
```

Android

要从数据集中删除键，请如下所示使用 `remove` 方法：

```
dataset.remove("myKey");
```

iOS - Objective-C

要从数据集中删除键，请如下所示使用 `removeObjectForKey`：

```
[dataset removeObjectForKey:@"myKey"];
```

iOS - Swift

要从数据集中删除键，请如下所示使用 `removeObjectForKey`：

```
dataset.removeObjectForKey("myKey")
```

Unity

要从数据集中删除键，请如下所示使用 `Remove`：

```
dataset.Remove("myKey");
```

Xamarin

您可以使用 `Remove` 从数据集中删除键：

```
dataset.Remove("myKey");
```

使用同步存储同步本地数据

Android

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

```
dataset.synchronize(syncCallback);
```

synchronize 方法收到 SyncCallback 接口的实现，如下所述。

synchronizeOnConnectivity() 方法尝试在连接可用时进行同步。如果连接立即可用，则 synchronizeOnConnectivity() 的行为类似于 synchronize()。否则，它会监控连接更改，并在连接可用时立即执行同步。如果多次调用 synchronizeOnConnectivity()，则只会保持最近一次同步请求，并只会触发最近一次回调。如果数据集或回调收集到垃圾，则此方法不会执行同步，且不会触发回调。

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

iOS - Objective-C

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

synchronize 方法是异步的，它会返回 AWSTask 对象以处理响应：

```
[[dataset synchronize] continueWithBlock:^id(AWSTask *task) {
    if (task.isCancelled) {
        // Task cancelled.
    } else if (task.error) {
        // Error while executing task.
    } else {
        // Task succeeded. The data was saved in the sync store.
    }
    return nil;
}];
```

synchronizeOnConnectivity 方法尝试在设备具备连接时进行同步。首先，synchronizeOnConnectivity 将检查连接状态，如果设备处于在线状态，则立即调用 synchronize，并返回与此次尝试关联的 AWSTask 对象。

如果设备处于离线状态，synchronizeOnConnectivity 1) 计划在设备下次变为在线状态时进行同步；2) 返回一个无结果的 AWSTask。计划的同步仅在该数据集对象的生命周期内有效。如果在连接恢复之前退出应用程序，则数据不会进行同步。如果您希望在计划同步期间发生事件时收到通知，则必须添加在 AWSCognito 中找到的通知的观察者。

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

iOS - Swift

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 synchronize 方法：

synchronize 方法是异步的，它会返回 AWSTask 对象以处理响应：

```
dataset.synchronize().continueWith(block: { (task) -> AnyObject? in

            if task.isCancelled {
                // Task cancelled.
            } else if task.error != nil {
                // Error while executing task
            } else {
                // Task succeeded. The data was saved in the sync store.
            }
            return task
})
```

synchronizeOnConnectivity 方法尝试在设备具备连接时进行同步。首先，synchronizeOnConnectivity 将检查连接状态，如果设备处于在线状态，则立即调用 synchronize，并返回与此次尝试关联的 AWSTask 对象。

如果设备处于离线状态，synchronizeOnConnectivity 1) 计划在设备下次变为在线状态时进行同步；2) 返回一个无结果的 AWSTask 对象。计划的同步仅在该数据集对象的生命周期内有效。如果在连接恢复之前退出应用程序，则数据不会进行同步。如果您希望在计划同步期间发生事件时收到通知，则必须添加在 AWSCognito 中找到的通知的观察者。

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

JavaScript

`synchronize` 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 `synchronize` 方法：

```
dataset.synchronize();
```

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

Unity

synchronize 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 `synchronize` 方法：

```
dataset.Synchronize();
```

同步将以异步方式运行，最终会调用您可以在数据集中指定的几个回调之一。

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

Xamarin

`synchronize` 方法将本地缓存的数据与存储在 Amazon Cognito Sync 存储空间中的数据进行比较。从 Amazon Cognito Sync 存储空间中拉取远程更改；如果出现任何冲突，则调用冲突解决方法；设备上的更新值将推送到该服务。要同步数据集，请调用其 `synchronize` 方法：

```
dataset.SynchronizeAsync();
```

要了解有关数据集同步和不同回调的更多信息，请参阅处理事件回调。

处理事件回调

> ⚠ 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。

它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

作为 Amazon Cognito Sync 开发人员，您可以实施各种回调来处理不同的同步事件和场景。Android SDK 中的 `SyncCallback` 接口配置有关数据集同步的通知，包括成功下载数据集时的 `onSuccess()`、发生异常时的 `onFailure()`，以及为解决本地和远程数据之间冲突而使用的 `onConflict()`。

在 iOS SDK 中，您可以注册类似的通知（例如 `AWSCognitoDidStartSynchronizeNotification`），也可以设置像 `AWSCognitoRecordConflictHandler` 这样的处理程序来解决冲突。 JavaScript、Unity 和 Xamarin 平台具有类似的回调机制。当您实施这些回调时，您的应用程序可以妥善地处理在使用 Amazon Cognito Sync 时发生的各种同步事件和场景。

Android

SyncCallback 接口

通过实施 `SyncCallback` 接口，您可以在应用程序上接收有关数据集同步的通知。然后，您的应用程序可以在删除本地数据、合并未经身份验证的和经过身份验证的配置文件以及解决同步冲突方面制定有效决策。您应该实施接口所需的以下方法：

- `onSuccess()`
- `onFailure()`
- `onConflict()`
- `onDatasetDeleted()`
- `onDatasetsMerged()`

请注意，如果您不想指定所有回调，也可以使用类 `DefaultSyncCallback`，它会为所有回调提供默认的空实施。

onSuccess

从同步存储空间成功下载数据集后，将触发 `onSuccess()` 回调。

```
@Override
```

```
public void onSuccess(Dataset dataset, List<Record> newRecords) {
}
```

onFailure

如果同步过程中出现异常，则会调用 onFailure()。

```
@Override
public void onFailure(DataStorageException dse) {
}
```

onConflict

如果在本地存储和同步存储空间修改同一键，则会产生冲突。`onConflict()` 方法可处理冲突解决方法。如果您没有实施此方法，则 Amazon Cognito Sync 客户端将默认为使用最近的更改。

```
@Override
public boolean onConflict(Dataset dataset, final List<SyncConflict> conflicts) {
    List<Record> resolvedRecords = new ArrayList<Record>();
    for (SyncConflict conflict : conflicts) {
        /* resolved by taking remote records */
        resolvedRecords.add(conflict.resolveWithRemoteRecord());

        /* alternately take the local records */
        // resolvedRecords.add(conflict.resolveWithLocalRecord());

        /* or customer logic, say concatenate strings */
        // String newValue = conflict.getRemoteRecord().getValue()
        //     + conflict.getLocalRecord().getValue();
        // resolvedRecords.add(conflict.resolveWithValue(newValue);
    }
    dataset.resolve(resolvedRecords);

    // return true so that synchronize() is retried after conflicts are resolved
    return true;
}
```

onDatasetDeleted

删除数据集后，Amazon Cognito 客户端将使用 `SyncCallback` 接口确认是否还应删除本地缓存的数据集副本。实施 `onDatasetDeleted()` 方法，以告知客户端开发工具包该如何处理本地数据。

```
@Override
public boolean onDatasetDeleted(Dataset dataset, String datasetName) {
    // return true to delete the local copy of the dataset
    return true;
}
```

onDatasetMerged

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `onDatasetsMerged()` 方法向应用程序发送有关合并的通知：

```
@Override
public boolean onDatasetsMerged(Dataset dataset, List<String> datasetNames) {
    // return false to handle Dataset merge outside the synchronization callback
    return false;
}
```

iOS - Objective-C

同步通知

在同步调用过程中，Amazon Cognito 客户端将发出大量 `NSNotification` 事件。您可以进行注册，以通过标准 `NSNotificationCenter` 监控这些通知：

```
[NSNotificationCenter defaultCenter]
  addObserver:self
  selector:@selector(myNotificationHandler:)
  name:NOTIFICATION_TYPE
  object:nil];
```

Amazon Cognito 支持五种通知类型，如下所列。

AWSCognitoDidStartSynchronizeNotification

同步操作开始时调用。`userInfo` 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidEndSynchronizeNotification

在同步操作完成 (无论是否成功) 时调用。`userInfo` 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidFailToSynchronizeNotification

同步操作失败时调用。`userInfo` 包含主数据集 (即正在进行同步的数据集的名称) 和关键错误 (包含导致失败的错误)。

AWSCognitoDidChangeRemoteValueNotification

本地更改成功推送到 Amazon Cognito 时调用。`userInfo`将包含密钥数据集（即正在同步的数据集的名称）和包含已推送 NSArray 的记录密钥的密钥。

AWSCognitoDidChangeLocalValueFromRemoteNotification

本地值由于同步操作而更改时调用。`userInfo`将包含密钥数据集（即正在同步的数据集的名称）和包含已更改记录密钥 NSArray 的密钥。

冲突解决方法处理程序

在同步操作过程中，如果在本地存储和同步存储空间修改同一键，则会产生冲突。如果您尚未设置冲突解决方法处理程序，则 Amazon Cognito 将默认为选择最近的更新。

通过实现和分配， AWSCognitoRecordConflictHandler 您可以更改默认的冲突解决方案。AWSCognito冲突输入参数冲突包含 AWSCognito本地缓存数据和同步存储中冲突记录的 Record 对象。使用 “ AWSCognito冲突”，您可以使用本地记录：[冲突记录]、远程 resolveWithLocal记录：[冲突 resolveWithRemote记录] 或全新的值：[冲突:value] 来解决冲突 resolveWithValue。此方法返回无将阻止同步继续进行，并且下一次同步过程开始时会再次出现冲突。

您可以在客户端层面设置冲突解决方法处理程序：

```
client.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // always choose local changes
    return [conflict resolveWithLocalRecord];
};
```

或在数据集层面：

```
dataset.conflictHandler = ^AWSCognitoResolvedConflict* (NSString *datasetName,
 AWSCognitoConflict *conflict) {
    // override and always choose remote changes
    return [conflict resolveWithRemoteRecord];
};
```

数据集删除处理程序

删除数据集后，Amazon Cognito 客户端将使用 AWSCognitoDatasetDeletedHandler 来确认是否还应删除本地缓存的数据集副本。如果未实施 AWSCognitoDatasetDeletedHandler，则将自动清除本地数据。如果您希望在清除前保留本地数据的副本或保留本地数据，则实施 AWSCognitoDatasetDeletedHandler。

您可以在客户端层面设置数据集删除处理程序：

```
client.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // make a backup of the data if you choose
    ...
    // delete the local data (default behavior)
    return YES;
};
```

或在数据集层面：

```
dataset.datasetDeletedHandler = ^BOOL (NSString *datasetName) {
    // override default and keep the local data
    return NO;
};
```

数据集合并处理程序

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 DatasetMergeHandler 向应用程序发送有关合并的通知。该处理程序将收到根数据集名称以及标记为根数据集合并的数据集名称数组。

如果未实施 DatasetMergeHandler，这些数据集将被忽略，但将继续占用最多 20 个身份总数据集中的空间。

您可以在客户端层面设置数据集合并处理程序：

```
client.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
       [merged clear];
```

```
        [merged synchronize];
    }
};
```

或在数据集层面：

```
dataset.datasetMergedHandler = ^(NSString *datasetName, NSArray *datasets) {
    // Blindly delete the datasets
    for (NSString *name in datasets) {
        AWSCognitoDataset *merged = [[AWSCognito defaultCognito]
 openOrCreateDataset:name];
        // do something with the data if it differs from existing dataset
        ...
        // now delete it
        [merged clear];
        [merged synchronize];
    }
};
```

iOS - Swift

同步通知

在同步调用过程中，Amazon Cognito 客户端将发出大量 `NSNotification` 事件。您可以进行注册，以通过标准 `NSNotificationCenter` 监控这些通知：

```
NSNotificationCenter.defaultCenter().addObserver(observer: self,
   selector: "myNotificationHandler",
   name:NOTIFICATION_TYPE,
   object:nil)
```

Amazon Cognito 支持五种通知类型，如下所列。

AWSCognitoDidStartSynchronizeNotification

同步操作开始时调用。`userInfo` 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidEndSynchronizeNotification

在同步操作完成 (无论是否成功) 时调用。`userInfo` 包含主数据集，即正在进行同步的数据集的名称。

AWSCognitoDidFailToSynchronizeNotification

同步操作失败时调用。`userInfo` 包含主数据集 (即正在进行同步的数据集的名称) 和关键错误 (包含导致失败的错误)。

AWSCognitoDidChangeRemoteValueNotification

本地更改成功推送到 Amazon Cognito 时调用。`userInfo`将包含密钥数据集（即正在同步的数据集的名称）和包含已推送 NSArray 的记录密钥的密钥。

AWSCognitoDidChangeLocalValueFromRemoteNotification

本地值由于同步操作而更改时调用。`userInfo`将包含密钥数据集（即正在同步的数据集的名称）和包含已更改记录密钥 NSArray 的密钥。

冲突解决方法处理程序

在同步操作过程中，如果在本地存储和同步存储空间修改同一键，则会产生冲突。如果您尚未设置冲突解决方法处理程序，则 Amazon Cognito 将默认为选择最近的更新。

通过实施和分配 `AWSCognitoRecordConflictHandler`，您可以更改默认的冲突解决方法。`AWSCognitoConflict` 输入参数冲突包含本地缓存数据和同步存储空间中冲突记录的 `AWSCognitoRecord` 对象。使用`AWSCognitoConflict`可以解决与本地记录：[冲突记录]、远程 resolveWithLocal记录：[冲突 resolveWithRemote记录] 或全新值：[冲突:value] 的冲突 resolveWithValue。此方法返回无将阻止同步继续进行，并且下一次同步过程开始时会再次出现冲突。

您可以在客户端层面设置冲突解决方法处理程序：

```
client.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
}
```

或在数据集层面：

```
dataset.conflictHandler = {
    (datasetName: String?, conflict: AWSCognitoConflict?) ->
 AWSCognitoResolvedConflict? in
    return conflict.resolveWithLocalRecord()
```

```
}
```

数据集删除处理程序

删除数据集后，Amazon Cognito 客户端将使用 `AWSCognitoDatasetDeletedHandler` 来确认是否还应删除本地缓存的数据集副本。如果未实施 `AWSCognitoDatasetDeletedHandler`，则将自动清除本地数据。如果您希望在清除前保留本地数据的副本或保留本地数据，则实施 `AWSCognitoDatasetDeletedHandler`。

您可以在客户端层面设置数据集删除处理程序：

```
client.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

或在数据集层面：

```
dataset.datasetDeletedHandler = {
      (datasetName: String!) -> Bool in
      // make a backup of the data if you choose
      ...
      // delete the local data (default behaviour)
      return true
}
```

数据集合并处理程序

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `DatasetMergeHandler` 向应用程序发送有关合并的通知。该处理程序将收到根数据集名称以及标记为根数据集合并的数据集名称数组。

如果未实施 `DatasetMergeHandler`，这些数据集将被忽略，但将继续占用最多 20 个身份总数据集中的空间。

您可以在客户端层面设置数据集合并处理程序：

```
client.datasetMergedHandler = {
```

```
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            merged.clear()
            merged.synchronize()
        }
    }
}
```

或在数据集层面：

```
dataset.datasetMergedHandler = {
    (datasetName: String!, datasets: [AnyObject]!) -> Void in
    for nameObject in datasets {
        if let name = nameObject as? String {
            let merged = AWSCognito.defaultCognito().openOrCreateDataset(name)
            // do something with the data if it differs from existing dataset
              ...
              // now delete it
            merged.clear()
            merged.synchronize()
        }
    }
}
```

JavaScript

同步回调

在数据集上执行 synchronize() 时，您可以有选择性地指定用于处理以下各种状态的回调：

```
dataset.synchronize({

   onSuccess: function(dataset, newRecords) {
      //...
   },

   onFailure: function(err) {
      //...
   },
```

```
   onConflict: function(dataset, conflicts, callback) {
      //...
   },

   onDatasetDeleted: function(dataset, datasetName, callback) {
      //...
   },

  onDatasetMerged: function(dataset, datasetNames, callback) {
      //...
  }

});
```

onSuccess()

从同步存储空间成功更新数据集后，将触发 `onSuccess()` 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
onSuccess: function(dataset, newRecords) {
   console.log('Successfully synchronized ' + newRecords.length + ' new records.');
}
```

onFailure()

如果同步过程中出现异常，则会调用 `onFailure()`。如果您未定义回调，则同步失败后将无提示。

```
onFailure: function(err) {
   console.log('Synchronization failed.');
   console.log(err);
}
```

onConflict()

如果在本地存储和同步存储空间修改同一键，则会产生冲突。`onConflict()` 方法可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
onConflict: function(dataset, conflicts, callback) {

   var resolved = [];
```

```
   for (var i=0; i<conflicts.length; i++) {

      // Take remote version.
      resolved.push(conflicts[i].resolveWithRemoteRecord());

      // Or... take local version.
      // resolved.push(conflicts[i].resolveWithLocalRecord());

      // Or... use custom logic.
      // var newValue = conflicts[i].getRemoteRecord().getValue() +
 conflicts[i].getLocalRecord().getValue();
      // resolved.push(conflicts[i].resovleWithValue(newValue);

   }

   dataset.resolve(resolved, function() {
      return callback(true);
   });

   // Or... callback false to stop the synchronization process.
   // return callback(false);

}
```

onDatasetDeleted()

删除数据集后，Amazon Cognito 客户端将使用 `onDatasetDeleted()` 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
onDatasetDeleted: function(dataset, datasetName, callback) {

   // Return true to delete the local copy of the dataset.
   // Return false to handle deleted datasets outside the synchronization callback.

   return callback(true);

}
```

onDatasetMerged()

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `onDatasetsMerged()` 回调向应用程序发送有关合并的通知。

```
onDatasetMerged: function(dataset, datasetNames, callback) {

   // Return true to continue the synchronization process.
   // Return false to handle dataset merges outside the synchronization callback.

   return callback(false);

}
```

Unity

打开或创建数据集之后，您可以为其设置不同的回调，以在使用 Synchronize 方法时触发。以下就是将回调注册到数据集的方法：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

请注意，`SyncSuccess` 和 `SyncFailure` 使用 `+=` 而不是 `=`，因此您可以向其订阅多个回调。

OnSyncSuccess

从云成功更新数据集后，将触发 `OnSyncSuccess` 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
private void HandleSyncSuccess(object sender, SyncSuccessEvent e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

如果同步过程中出现异常，则会调用 `OnSyncFailure`。如果您未定义回调，则同步失败后将无提示。

```
private void HandleSyncFailure(object sender, SyncFailureEvent e)
{
```

```
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Debug.Log("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Debug.Log("Sync failed");
    }
    // Handle the error
    Debug.LogException(e.Exception);
}
```

OnSyncConflict

如果在本地存储和同步存储空间修改同一键，则会产生冲突。`OnSyncConflict` 回调可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Debug.LogWarning("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Debug.LogWarning("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

删除数据集后，Amazon Cognito 客户端将使用 `OnDatasetDeleted` 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
private bool HandleDatasetDeleted(Dataset dataset)
  {
      Debug.Log(dataset.Metadata.DatasetName + " Dataset has been deleted");
      // Do clean up if necessary
      // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
      return true;
  }
```

OnDatasetMerged

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `OnDatasetsMerged` 回调向应用程序发送有关合并的通知。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);
        //Lambda function to delete the dataset after fetching it
        EventHandler<SyncSuccessEvent> lambda;
        lambda = (object sender, SyncSuccessEvent e) => {
            ICollection<string> existingValues = localDataset.GetAll().Values;
            ICollection<string> newValues = mergedDataset.GetAll().Values;

            //Implement your merge logic here

            mergedDataset.Delete(); //Delete the dataset locally
            mergedDataset.OnSyncSuccess -= lambda; //We don't want this callback to be
 fired again
            mergedDataset.OnSyncSuccess += (object s2, SyncSuccessEvent e2) => {
                localDataset.Synchronize(); //Continue the sync operation that was
 interrupted by the merge
            };
            mergedDataset.Synchronize(); //Synchronize it as deleted, failing to do so
 will leave us in an inconsistent state
        };
        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.Synchronize(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
```

```
}
```

Xamarin

打开或创建数据集之后，您可以为其设置不同的回调，以在使用 Synchronize 方法时触发。以下就是将回调注册到数据集的方法：

```
dataset.OnSyncSuccess += this.HandleSyncSuccess;
dataset.OnSyncFailure += this.HandleSyncFailure;
dataset.OnSyncConflict = this.HandleSyncConflict;
dataset.OnDatasetMerged = this.HandleDatasetMerged;
dataset.OnDatasetDeleted = this.HandleDatasetDeleted;
```

请注意，`SyncSuccess` 和 `SyncFailure` 使用 `+=` 而不是 `=`，因此您可以向其订阅多个回调。

OnSyncSuccess

从云成功更新数据集后，将触发 `OnSyncSuccess` 回调。如果您未定义回调，则同步成功完成后将保持静默状态。

```
private void HandleSyncSuccess(object sender, SyncSuccessEventArgs e)
{
    // Continue with your game flow, display the loaded data, etc.
}
```

OnSyncFailure

如果同步过程中出现异常，则会调用 `OnSyncFailure`。如果您未定义回调，则同步失败后将无提示。

```
private void HandleSyncFailure(object sender, SyncFailureEventArgs e)
{
    Dataset dataset = sender as Dataset;
    if (dataset.Metadata != null) {
        Console.WriteLine("Sync failed for dataset : " + dataset.Metadata.DatasetName);
    } else {
        Console.WriteLine("Sync failed");
    }
}
```

OnSyncConflict

如果在本地存储和同步存储空间修改同一键，则会产生冲突。`OnSyncConflict` 回调可处理冲突解决方法。如果您未实施此方法，则在发生冲突时，同步将中止。

```
private bool HandleSyncConflict(Dataset dataset, List < SyncConflict > conflicts)
{
  if (dataset.Metadata != null) {
    Console.WriteLine("Sync conflict " + dataset.Metadata.DatasetName);
  } else {
    Console.WriteLine("Sync conflict");
  }
  List < Amazon.CognitoSync.SyncManager.Record > resolvedRecords = new List <
 Amazon.CognitoSync.SyncManager.Record > ();
  foreach(SyncConflict conflictRecord in conflicts) {
    // SyncManager provides the following default conflict resolution methods:
    //      ResolveWithRemoteRecord - overwrites the local with remote records
    //      ResolveWithLocalRecord - overwrites the remote with local records
    //      ResolveWithValue - to implement your own logic
    resolvedRecords.Add(conflictRecord.ResolveWithRemoteRecord());
  }
  // resolves the conflicts in local storage
  dataset.Resolve(resolvedRecords);
  // on return true the synchronize operation continues where it left,
  //      returning false cancels the synchronize operation
  return true;
}
```

OnDatasetDeleted

删除数据集后，Amazon Cognito 客户端将使用 `OnDatasetDeleted` 回调决定是否还应删除本地缓存的数据集副本。默认情况下，不会删除该数据集。

```
private bool HandleDatasetDeleted(Dataset dataset)
{
    Console.WriteLine(dataset.Metadata.DatasetName + " Dataset has been deleted");
    // Do clean up if necessary
    // returning true informs the corresponding dataset can be purged in the local
 storage and return false retains the local dataset
    return true;
}
```

OnDatasetMerged

当两个以前未连接的身份链接在一起后，它们的所有数据集都将合并。可通过 `OnDatasetsMerged` 回调向应用程序发送有关合并的通知。

```
public bool HandleDatasetMerged(Dataset localDataset, List<string> mergedDatasetNames)
{
    foreach (string name in mergedDatasetNames)
    {
        Dataset mergedDataset = syncManager.OpenOrCreateDataset(name);

            //Implement your merge logic here

        mergedDataset.OnSyncSuccess += lambda;
        mergedDataset.SynchronizeAsync(); //Asnchronously fetch the dataset
    }

    // returning true allows the Synchronize to continue and false stops it
    return false;
}
```

实施推送同步

> 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito 会自动跟踪身份和设备之间的关联。使用推送同步功能，您可以确保在身份数据发生更改后向给定身份的每个实例发送通知。推送同步可以确保，无论特定身份的同步存储数据何时发生更改，与该身份关联的所有设备都会收到一个静音推送通知，通知它们所发生的更改。

> **Note**
>
> Unity 或 Xamarin 不支持推送同步。 JavaScript

您必须首先设置用于推送同步的账户，并在 Amazon Cognito 控制台中启用推送同步，然后才可使用推送同步。

创建 Amazon Simple Notification Service（Amazon SNS）应用程序

为支持的平台创建并配置 Amazon SNS 应用程序，如 SNS 开发人员指南中所述。

在 Amazon Cognito 控制台中启用推送同步

您可以通过 Amazon Cognito 控制台启用推送同步。从控制台主页：

1. 单击您需要启用推送同步的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在Dashboard（控制面板）页的右上角，单击 Manage Identity Pools（管理身份池）。此时将显示 Federated Identities（联合身份）页。
3. 向下滚动并单击 Push synchronization（推送同步）以将其展开。
4. 在 Service role（服务角色）下拉菜单中，选择授予 Cognito 发送 SNS 通知的权限的 IAM 角色。在 Amazon IAM 控制台中，单击 Create role (创建角色) 以创建或修改与您身份池关联的角色。
5. 选择一个平台应用程序，然后单击 Save Changes（保存更改）。
6. 为应用程序授予 SNS 访问权限

在 Amazon Identity and Access Management 控制台中，将您的 IAM 角色配置为拥有完全的 Amazon SNS 访问权限，或者创建一个具有完整 Amazon SNS 访问权限的新角色。以下示例角色信任策略授予 Amazon Cognito Sync 有限代入 IAM 角色的能力。Amazon Cognito Sync 只能在代表 `aws:SourceArn` 条件中的身份池和 `aws:SourceAccount` 条件中的账户时才能代入该角色。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "Service": "cognito-sync.amazonaws.com"
            },
            "Action": "sts:AssumeRole",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "123456789012"
                },
                "ArnLike": {
                    "AWS:SourceArn": "arn:aws:cognito-identity:us-
east-1:123456789012:identitypool/us-east-1:177a950c-2c08-43f0-9983-28727EXAMPLE"
```

```
                }
            }
        }
    ]
}
```

要了解有关 IAM 角色的更多信息，请参阅角色（委托和联合）。

在您的应用程序中使用推送同步：Android

您的应用程序需要导入 Google Play 服务。您可以通过 Android SDK Manager 下载最新版本的 Google Play 开发工具包。按照 Android 有关 Android 实施的文档注册您的应用程序并接收来自 GCM 的注册 ID。收到注册 ID 之后，您需要向 Amazon Cognito 注册设备，如以下代码段所示：

```
String registrationId = "MY_GCM_REGISTRATION_ID";
try {
    client.registerDevice("GCM", registrationId);
} catch (RegistrationFailedException rfe) {
    Log.e(TAG, "Failed to register device for silent sync", rfe);
} catch (AmazonClientException ace) {
    Log.e(TAG, "An unknown error caused registration for silent sync to fail", ace);
}
```

现在，您可以订阅设备以接收来自特定数据集的更新：

```
Dataset trackedDataset = client.openOrCreateDataset("myDataset");
if (client.isDeviceRegistered()) {
    try {
        trackedDataset.subscribe();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

要停止接收来自数据集的推送通知，只需调用 unsubscribe 方法即可。要订阅 `CognitoSyncManager` 对象中的所有数据集（或特定子集），请使用 `subscribeAll()`：

```
if (client.isDeviceRegistered()) {
```

```
    try {
        client.subscribeAll();
    } catch (SubscribeFailedException sfe) {
        Log.e(TAG, "Failed to subscribe to datasets", sfe);
    } catch (AmazonClientException ace) {
        Log.e(TAG, "An unknown error caused the subscription to fail", ace);
    }
}
```

在您的 Android BroadcastReceiver 对象实现中，您可以检查修改后的数据集的最新版本，并决定您的应用是否需要再次同步：

```
@Override
public void onReceive(Context context, Intent intent) {

    PushSyncUpdate update = client.getPushSyncUpdate(intent);

    // The update has the source (cognito-sync here), identityId of the
    // user, identityPoolId in question, the non-local sync count of the
    // data set and the name of the dataset. All are accessible through
    // relevant getters.

    String source = update.getSource();
    String identityPoolId = update.getIdentityPoolId();
    String identityId = update.getIdentityId();
    String datasetName = update.getDatasetName;
    long syncCount = update.getSyncCount;

    Dataset dataset = client.openOrCreateDataset(datasetName);

    // need to access last sync count. If sync count is less or equal to
    // last sync count of the dataset, no sync is required.

    long lastSyncCount = dataset.getLastSyncCount();
    if (lastSyncCount < syncCount) {
        dataset.synchronize(new SyncCallback() {
            // ...
        });
    }

}
```

推送通知负载中提供以下键：

- `source`：cognito-sync。这可以作为通知之间的区分因素。
- `identityPoolId`：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。
- `identityId`：池中的身份 ID。
- `datasetName`：已更新的数据集的名称。这是为了调用 openOrCreate数据集而提供的。
- `syncCount`：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

在您的应用程序中使用推送同步：iOS – Objective-C

要获取应用程序的设备令牌，请参阅 Apple 有关注册远程通知的文档。收到作为 NSData 对象的设备令牌后 APNs，您需要使用同步客户端的`registerDevice:`方法向 Amazon Cognito 注册设备，如下所示：

```
AWSCognito *syncClient = [AWSCognito defaultCognito];
    [[syncClient registerDevice: devToken] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to registerDevice: %@", task.error);
        } else {
            NSLog(@"Successfully registered device with id: %@", task.result);
        }
        return nil;
      }
    ];
```

在调试模式下，您的设备将在 APNs 沙盒中注册；在发布模式下，设备将在沙盒中注册。 APNs要接收来自特定数据集的更新，请使用 `subscribe` 方法：

```
[[[syncClient openOrCreateDataset:@"MyDataset"] subscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to dataset: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

要停止接收来自数据集的推送通知，只需调用 unsubscribe 方法即可。

```
[[[syncClient openOrCreateDataset:@”MyDataset”] unsubscribe]
 continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to unsubscribe from dataset: %@", task.error);
        } else {
            NSLog(@"Successfully unsubscribed from dataset: %@", task.result);
        }
        return nil;
      }
    ];
```

要订阅 AWSCognito 对象中的所有数据集，请调用 subscribeAll：

```
[[syncClient subscribeAll] continueWithBlock:^id(AWSTask *task) {
        if(task.error){
            NSLog(@"Unable to subscribe to all datasets: %@", task.error);
        } else {
            NSLog(@"Successfully subscribed to all datasets: %@", task.result);
        }
        return nil;
      }
    ];
```

在调用 subscribeAll 之前，请确保在每个数据集上至少同步一次，以便数据集存在于服务器上。

要对推送通知做出反应，您需要在应用程序委托上实施 didReceiveRemoteNotification 方法：

```
- (void)application:(UIApplication *)application didReceiveRemoteNotification:
(NSDictionary *)userInfo
    {
        [[NSNotificationCenter defaultCenter]
 postNotificationName:@"CognitoPushNotification" object:userInfo];
    }
```

如果您使用通知处理程序发布通知，则可以在您拥有数据集句柄的应用程序中的其他位置响应通知。如果您按照如下方式订阅通知...

```
[[NSNotificationCenter defaultCenter] addObserver:self
 selector:@selector(didReceivePushSync:)
```

```
    name: :@"CognitoPushNotification" object:nil];
```

...则可以按照如下所示处理通知：

```
- (void)didReceivePushSync:(NSNotification*)notification
    {
        NSDictionary * data = [(NSDictionary *)[notification object]
 objectForKey:@"data"];
        NSString * identityId = [data objectForKey:@"identityId"];
        NSString * datasetName = [data objectForKey:@"datasetName"];
        if([self.dataset.name isEqualToString:datasetName] && [self.identityId
 isEqualToString:identityId]){
            [[self.dataset synchronize] continueWithBlock:^id(AWSTask *task) {
                if(!task.error){
                    NSLog(@"Successfully synced dataset");
                }
                return nil;
            }];
        }
    }
```

推送通知负载中提供以下键：

- `source`：cognito-sync。这可以作为通知之间的区分因素。
- `identityPoolId`：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。
- `identityId`：池中的身份 ID。
- `datasetName`：已更新的数据集的名称。这可用于 `openOrCreateDataset` 调用。
- `syncCount`：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

在您的应用程序中使用推送同步：iOS – Swift

要获取应用程序的设备令牌，请参阅 Apple 有关注册远程通知的文档。收到作为 NSData 对象的设备令牌后 APNs，您需要使用同步客户端的 RegisterDevice: 方法向 Amazon Cognito 注册设备，如下所示：

```
let syncClient = AWSCognito.default()
```

```
syncClient.registerDevice(devToken).continueWith(block: { (task: AWSTask!) ->
 AnyObject! in
    if (task.error != nil) {
        print("Unable to register device: " + task.error.localizedDescription)

    } else {
        print("Successfully registered device with id: \(task.result)")
    }
    return task
})
```

在调试模式下，您的设备将在 APNs 沙盒中注册；在发布模式下，设备将在沙盒中注册。 APNs要接收来自特定数据集的更新，请使用 `subscribe` 方法：

```
syncClient.openOrCreateDataset("MyDataset").subscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to subscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to dataset: \(task.result)")
  }
  return task
})
```

要停止接收来自数据集的推送通知，请调用 `unsubscribe` 方法：

```
syncClient.openOrCreateDataset("MyDataset").unsubscribe().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
  if (task.error != nil) {
      print("Unable to unsubscribe to dataset: " + task.error.localizedDescription)

  } else {
      print("Successfully unsubscribed to dataset: \(task.result)")
  }
  return task
})
```

要订阅 `AWSCognito` 对象中的所有数据集，请调用 `subscribeAll`：

```
syncClient.openOrCreateDataset("MyDataset").subscribeAll().continueWith(block: { (task:
 AWSTask!) -> AnyObject! in
```

```
  if (task.error != nil) {
      print("Unable to subscribe to all datasets: " + task.error.localizedDescription)

  } else {
      print("Successfully subscribed to all datasets: \(task.result)")
  }
  return task
})
```

在调用 `subscribeAll` 之前，请确保在每个数据集上至少同步一次，以便数据集存在于服务器上。

要对推送通知做出反应，您需要在应用程序委托上实施 `didReceiveRemoteNotification` 方法：

```
func application(application: UIApplication, didReceiveRemoteNotification userInfo:
 [NSObject : AnyObject],
  fetchCompletionHandler completionHandler: (UIBackgroundFetchResult) -> Void) {

 NSNotificationCenter.defaultCenter().postNotificationName("CognitoPushNotification",
 object: userInfo)
})
```

如果您使用通知处理程序发布通知，则可以在您拥有数据集句柄的应用程序中的其他位置响应通知。如果您按照如下方式订阅通知...

```
NSNotificationCenter.defaultCenter().addObserver(observer:self,
   selector:"didReceivePushSync:",
   name:"CognitoPushNotification",
   object:nil)
```

...则可以按照如下所示处理通知：

```
func didReceivePushSync(notification: NSNotification) {
    if let data = (notification.object as! [String: AnyObject])["data"] as? [String:
 AnyObject] {
        let identityId = data["identityId"] as! String
        let datasetName = data["datasetName"] as! String

        if self.dataset.name == datasetName && self.identityId == identityId {
          dataset.synchronize().continueWithBlock {(task) -> AnyObject! in
              if task.error == nil {
                print("Successfully synced dataset")
              }
```

```
                return nil
            }
        }
    }
}
```

推送通知负载中提供以下键：

- `source`：cognito-sync。这可以作为通知之间的区分因素。
- `identityPoolId`：身份池 ID。这可用于验证或获取其他信息，但从接收方的角度来看，这并不是不可或缺的。
- `identityId`：池中的身份 ID。
- `datasetName`：已更新的数据集的名称。这可用于 `openOrCreateDataset` 调用。
- `syncCount`：远程数据集的同步计数。您可以使用此方法来确保本地数据集已过期，并且传入同步是新的。

实施 Amazon Cognito Sync 流

> 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Streams 让开发人员能够控制和了解他们存储在 Amazon Cognito 中的数据。现在，开发人员可以配置 Kinesis 流，以便在数据更新和同步时接收事件。Amazon Cognito 可以实时向您拥有的 Kinesis 流推送每个数据集更改。

使用 Amazon Cognito Streams，您可以将所有的同步数据移动到 Kinesis，然后将其流式传输到数据仓库工具（如 Amazon Redshift），供进一步分析。要了解有关 Kinesis 的更多信息，请参阅 Amazon Kinesis 入门。

配置流

您可以在 Amazon Cognito 控制台中设置 Amazon Cognito Streams。要在 Amazon Cognito 控制台中启用 Amazon Cognito Streams，您需要选择要将它发布到哪个 Kinesis 流，以及授权 Amazon Cognito 将事件放入选定流中的 IAM 角色。

从控制台主页：

1. 单击要为其设置 Amazon Cognito 流的身份池的名称。此时将显示身份池的 Dashboard（控制面板）页。
2. 在 Dashboard（控制面板）页的右上角，单击 Manage Identity Pools（管理身份池）。出现“Manage Federated Identities”（管理联合身份）页。
3. 向下滚动，并单击 Cognito Streams（Cognito 流），以将其展开。
4. 在 Stream name（流名称）下拉菜单中，选择一个现有 Kinesis 流的名称。或者，单击 Create stream（创建流）以创建一个流，输入流的名称和分片数量。要了解分片，以及获取有关如何估算流需要的分片数的帮助，请参阅 Kinesis 开发人员指南。
5. 在 Publish role (发布角色) 下拉菜单中，选择授予 Amazon Cognito 发布流的权限的 IAM 角色。在 Amazon IAM 控制台中，单击 Create role (创建角色) 以创建或修改与您身份池关联的角色。
6. 在 Stream status（流状态）下拉菜单中，选择 Enabled（启用）以启用流更新。单击 Save Changes（保存更改）。

成功配置 Amazon Cognito Streams 之后，此身份池中数据集的所有后续更新都会发送到此流中。

流内容

发送到流的每个记录都代表一次同步。以下是一个发送到流的记录示例：

```
{
    "identityPoolId": "Pool Id",
    "identityId": "Identity Id",
    "dataSetName": "Dataset Name",
    "operation": "(replace|remove)",
    "kinesisSyncRecords": [
        {
            "key": "Key",
            "value": "Value",
            "syncCount": 1,
            "lastModifiedDate": 1424801824343,
            "deviceLastModifiedDate": 1424801824343,
            "op": "(replace|remove)"
        },
        ...
    ],
    "lastModifiedDate": 1424801824343,
    "kinesisSyncRecordsURL": "S3Url",
```

```
    "payloadType": "(S3Url|Inline)",
    "syncCount": 1
}
```

对大于 Kinesis 最大有效负载大小 (1 MB) 的更新，Amazon Cognito 将包括预签名 Amazon S3 URL，其中包含更新的完整内容。

配置 Amazon Cognito Streams 之后，如果您删除 Kinesis 流或更改角色信任权限，使得 Amazon Cognito Sync 不再代入该角色，则您将禁用 Amazon Cognito 流。您必须重新创建 Kinesis 流或修复角色，然后必须重新启用此流。

批量发布

配置 Amazon Cognito Streams 之后，您能够对身份池中的现有数据执行批量发布操作。通过控制台或直接通过 API 启动批量发布操作之后，Amazon Cognito 会开始将此数据发布到接收更新的同一流中。

Amazon Cognito 不保证在使用批量发布操作时发送到流的数据具有唯一性。您可能会收到两个相同的更新，一个来自更新操作，一个属于批量发布。在处理来自流的记录时，请记住这一点。

要批量发布所有流，请执行“配置流”下的步骤 1-6，然后单击“Start bulk publish”（开始批量发布）。任何特定时间都只能有一个正在进行的批量发布操作，且每 24 小时只能有一次成功的批量发布请求。

使用 Amazon Cognito Events 自定义工作流

> ⚠ 如果您是 Amazon Cognito Sync 的新用户，请使用 Amazon AppSync。与 Amazon Cognito Sync 一样， Amazon AppSync 它是一项用于跨设备同步应用程序数据的服务。
> 它允许同步用户数据，如应用程序首选项或游戏状态。它还通过允许多个用户实时同步和协作处理共享的数据，来扩展这些功能。

Amazon Cognito Events 允许您执行 Amazon Lambda 函数以响应 Amazon Cognito 中的重要事件。当数据集得到同步时，Amazon Cognito 会引发同步触发事件。当用户更新数据时，您可以使用同步触发事件采取行动。该函数可以评估并有选择性地操作数据，然后，数据才会存储到云中并同步到用户的其他设备。这有利于在来自设备的数据同步到用户的其他设备之前对其进行验证，或者基于传入数据更新数据集中的其他值，如在玩家达到新级别时颁发奖励。

以下步骤将引导您设置每次 Amazon Cognito 数据集同步时都会执行的 Lambda 函数。

> **Note**
>
> 使用 Amazon Cognito Events 时，您只能使用从 Amazon Cognito 身份获取的凭证。如果您关联了 Lambda 函数，但`UpdateRecords`使用 Amazon 账户证书（开发者证书）调用，则不会调用您的 Lambda 函数。

在中创建函数 Amazon Lambda

要将 Lambda 与 Amazon Cognito 集成，您首先需要在 Lambda 中创建函数。为此，请执行以下操作：

在 Amazon Cognito 中选择 Lambda 函数

1. 打开 Lambda 控制台。
2. 单击“Create a Lambda function”（创建 Lambda 函数）。
3. 在选择蓝图屏幕上，搜索并选择 “cognito-sync-trigger。”
4. 在“Configure event sources”（配置事件源）屏幕上，将“Event source type”（事件源类型）设置保留为“Cognito Sync Trigger”，并选择您的身份池。单击“Next”（下一步）。

> **Note**
>
> 在控制台外配置 Amazon Cognito Sync 触发器时，您必须添加 Lambda 基于资源的权限才能允许 Amazon Cognito 调用该函数。您可以从 Lambda 控制台（参见使用基于资源的策略 Amazon Lambda）或使用 Lambda 操作添加此权限。AddPermission
> Lambda 基于资源的策略示例
> 以下 Amazon Lambda 基于资源的策略授予 Amazon Cognito 有限调用 Lambda 函数的能力。Amazon Cognito 只能在代表 `aws:SourceArn` 条件中的身份池和 `aws:SourceAccount` 条件中的账户时才能调用该函数。
>
> ```
> {
>  "Version": "2012-10-17",
>  "Id": "default",
>  "Statement": [
>  {
>  "Sid": "lambda-allow-cognito-my-function",
>  "Effect": "Allow",
>  "Principal": {
>  "Service": "cognito-sync.amazonaws.com"
> ```

```
            },
            "Action": "lambda:InvokeFunction",
            "Resource": "<your Lambda function ARN>",
            "Condition": {
                "StringEquals": {
                    "AWS:SourceAccount": "<your account number>"
                },
                "ArnLike": {
                    "AWS:SourceArn": "<your identity pool ARN>"
                }
            }
        }
    ]
}
```

5. 在“Configure function”（配置函数）屏幕上，输入函数的名称和描述。将“Runtime”（运行时）设置保留为“Node.js”。在我们的示例中保留原来的代码。默认示例没有更改正在同步的数据。它只记录发生了 Amazon Cognito 同步触发事件这一事实。将“Handler name”（处理程序名称）设置保留为“index.handler”。对于 Role (角色)，选择一个授权您的代码访问 Amazon Lambda的 IAM 角色。要修改角色，请参阅 IAM 控制台。将“Advanced”（高级）设置保留不变。单击“Next”（下一步）。
6. 在“Review”（查看）屏幕上，查看详细信息，并单击“Create function”（创建函数）。下一页面将显示您的新 Lambda 函数。

现在，Lambda 中已经写入了相应的函数，您需要选择该函数作为 Amazon Cognito Sync 触发事件的处理程序。以下步骤将指导您完成此过程。

从控制台主页：

1. 单击要为其设置 Amazon Cognito Events 的身份池的名称。此时将显示身份池的“Dashboard”（控制面板）页。
2. 在“Dashboard”（控制面板）页的右上角，单击“Manage Federated Identities”（管理联合身份）。出现“Manage Federated Identities”（管理联合身份）页。
3. 向下滚动，并单击“Cognito Events”（Cognito 事件），以将其展开。
4. 在 Sync Trigger (同步触发) 下拉菜单中，选择您希望在同步事件发生时触发的 Lambda 函数。
5. 单击“Save Changes”（保存更改）。

现在，您的 Lambda 函数将在每次数据集同步时执行。下一部分将介绍如何在数据同步时读取和修改函数中的数据。

为同步触发编写 Lambda 函数

同步触发器遵循服务提供商接口使用的编程模式。Amazon Cognito 将按照以下 JSON 格式向您的 Lambda 函数提供输入。

```
{
  "version": 2,
  "eventType": "SyncTrigger",
  "region": "us-east-1",
  "identityPoolId": "identityPoolId",
  "identityId": "identityId",
  "datasetName": "datasetName",
  "datasetRecords": {
    "SampleKey1": {
      "oldValue": "oldValue1",
      "newValue": "newValue1",
      "op": "replace"
    },
    "SampleKey2": {
      "oldValue": "oldValue2",
      "newValue": "newValue2",
      "op": "replace"
    },..
  }
}
```

Amazon Cognito 预计函数的返回值与输入格式相同。

编写同步触发器事件的函数时，应注意以下事项：

- 当 Amazon Cognito 在此期间调用您的 Lambda 函数时 UpdateRecords，该函数必须在 5 秒钟内做出响应。如果没有，则 Amazon Cognito Sync 服务将生成 `LambdaSocketTimeoutException` 异常。您无法增加此超时值。
- 如果您遇到了 `LambdaThrottledException` 异常，请再次尝试同步操作以更新记录。
- Amazon Cognito 提供数据集中出现的所有记录，以作为函数的输入。
- 应用程序用户更新中将 op 字段设置为 `replace` 的记录。删除的记录将 op 字段设置为 `remove`。
- 您可以修改任何记录，即使应用程序用户没有更新该记录也是如此。
- 除 datasetRecords 之外的所有字段均为只读。请勿更改它们。如果您更改这些字段，则无法更新记录。
- 要修改记录的值，只需更新该值并将 op 设置为 `replace`。

- 要删除记录，请将 op 设置为 `remove`，或将该值设置为空。
- 要添加记录，请将新记录添加到 datasetRecords 数组。
- Amazon Cognito 更新记录时，Amazon Cognito 会忽略响应中任何省略的记录。

示例 Lambda 函数

以下示例 Lambda 函数显示如何访问、修改和删除数据。

```
console.log('Loading function');

exports.handler = function(event, context) {
    console.log(JSON.stringify(event, null, 2));

    //Check for the event type
    if (event.eventType === 'SyncTrigger') {

        //Modify value for a key
        if('SampleKey1' in event.datasetRecords){
            event.datasetRecords.SampleKey1.newValue = 'ModifyValue1';
            event.datasetRecords.SampleKey1.op = 'replace';
        }

        //Remove a key
        if('SampleKey2' in event.datasetRecords){
            event.datasetRecords.SampleKey2.op = 'remove';
        }

        //Add a key
        if(!('SampleKey3' in event.datasetRecords)){
            event.datasetRecords.SampleKey3={'newValue':'ModifyValue3', 'op' :
 'replace'};
        }

    }
    context.done(null, event);
};
```

Amazon Cognito 中的安全性

云安全 Amazon 是重中之重。作为 Amazon 客户，您可以受益于专为满足大多数安全敏感型组织的要求而构建的数据中心和网络架构。

安全是双方共同承担 Amazon 的责任。责任共担模式将此描述为云的安全性和云中的安全性：

- 云安全 — Amazon 负责保护在 Amazon 云中运行 Amazon 服务的基础架构。 Amazon 还为您提供可以安全使用的服务。作为的一部分，第三方审计师定期测试和验证我们安全的有效性。要了解适用于 Amazon Cognito 的合规计划，请参阅按合规计划提供的划分的范围内的Amazon 服务。
- 云端安全-您的责任由您使用的 Amazon 服务决定。您还需要对其他因素负责，包括您的数据的敏感性、您公司的要求以及适用的法律法规。

该文档帮助您了解如何在使用 Amazon Cognito 时应用责任共担模式。它说明了如何配置 Amazon Cognito 以实现您的安全性和合规性目标。您还将学习如何使用其他 Amazon 服务来帮助您监控和保护您的 Amazon Cognito 资源。

内容

- Amazon Cognito 中的数据保护
- 适用于 Amazon Cognito 的 Identity and Access Management
- Amazon Cognito 中的日志记录和监控
- Amazon Cognito 的合规性验证
- Amazon Cognito 中的恢复能力
- Amazon Cognito 中的基础设施安全性
- Amazon Cognito 用户池中的配置和漏洞分析
- Amazon 亚马逊 Cognito 的托管策略

Amazon Cognito 中的数据保护

分担责任模型 Amazon 分适用于亚马逊 Cognito（Amazon Cognito）中的数据保护。如本模型所述 Amazon ，负责保护运行所有 Amazon 云的全球基础架构。您负责维护对托管在此基础结构上的内容的控制。此内容包括您使用的 Amazon 服务的安全配置和管理任务。有关数据隐私的更多信息，请参阅数据隐私常见问题。

出于数据保护目的，我们建议您保护 Amazon 账户凭证并使用 Amazon Identity and Access Management (IAM) 设置个人用户账户。这仅向每个用户授予履行其工作职责所需的权限。我们还建议您通过以下方式保护数据：

- 对每个账户使用多重身份验证（MFA）。
- 使用 SSL/TLS 与资源通信。 Amazon
- 使用设置 API 和用户活动日志 Amazon CloudTrail。
- 使用 Amazon 加密解决方案以及 Amazon 服务中的所有默认安全控制。
- 使用高级托管安全服务（例如 Amazon Macie），它有助于发现和保护存储在 Amazon S3 中的个人数据。

我们强烈建议您切勿将敏感的可识别信息（例如您客户的账号）放入自由格式字段（例如名称字段）。这包括您使用控制台、API 或使用 Amazon Cognito 或其他 Amazon 服务时。 Amazon CLI Amazon SDKs您输入到 Amazon Cognito 或其他服务中的任何数据都可能被选取以包含在诊断日志中。当您向外部服务器提供网址时，请勿在网址中包含凭证信息来验证您对该服务器的请求。

数据加密

数据加密通常分为两类：静态加密和传输中加密。

静态加密

Amazon Cognito 中的数据按照行业标准进行静态加密。

传输中加密

作为一项托管服务，Amazon Cognito 受到 Amazon 全球网络安全的保护。有关 Amazon 安全服务以及如何 Amazon 保护基础设施的信息，请参阅Amazon 云安全。要使用基础设施安全的最佳实践来设计您的 Amazon 环境，请参阅 S Amazon ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 Amazon 已发布的 API 调用通过网络访问 Amazon Cognito。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service（Amazon STS）生成临时安全凭证来对请求进行签名。

Amazon Cognito 用户池和身份池具有经过 IAM 身份验证、未经身份验证和经过令牌授权的 API 操作。未经身份验证和经过令牌授权的 API 操作旨在由您的客户（即您的应用程序的最终用户）使用。未经身份验证和经过令牌授权的 API 操作会进行静态加密和传输中加密。有关更多信息，请参阅 按授权模型分组的 API 操作列表。

> **Note**
>
> Amazon Cognito 在内部加密您的内容，并且不支持客户提供的密钥。

适用于 Amazon Cognito 的 Identity and Access Management

Amazon Identity and Access Management (IAM) Amazon Web Services 服务 可帮助管理员安全地控制对 Amazon 资源的访问权限。IAM 管理员控制谁可以通过身份验证（登录）和授权（具有权限）来使用 Amazon Cognito 资源。您可以使用 IAM Amazon Web Services 服务 ，无需支付额外费用。

主题

- 受众
- 使用身份进行身份验证
- 使用策略管理访问
- Amazon Cognito 如何与 IAM 配合使用
- 适用于 Amazon Cognito 的基于身份的策略示例
- Amazon Cognito 身份和访问问题排查
- 对 Amazon Cognito 使用服务相关角色

受众

您的使用方式 Amazon Identity and Access Management (IAM) 会有所不同，具体取决于您在 Amazon Cognito 中所做的工作。

服务用户 – 如果您使用 Amazon Cognito 服务来完成任务，则您的管理员会为您提供所需的凭证和权限。当您使用更多 Amazon Cognito 功能来完成工作时，您可能需要更多权限。了解如何管理访问权限

有助于您向管理员请求适合的权限。如果您无法访问 Amazon Cognito 中的一项功能，请参阅Amazon Cognito 身份和访问问题排查。

服务管理员 – 如果您在公司负责管理 Amazon Cognito 资源，则您可能具有 Amazon Cognito 的完全访问权限。您有责任确定您的服务用户应访问哪些 Amazon Cognito 功能和资源。然后，您必须向 IAM 管理员提交请求以更改服务用户的权限。请查看该页面上的信息以了解 IAM 的基本概念。要了解有关您的公司如何将 IAM 与 Amazon Cognito 结合使用的更多信息，请参阅Amazon Cognito 如何与 IAM 配合使用。

IAM 管理员 - 如果您是 IAM 管理员，您可能希望了解有关如何编写策略以管理对 Amazon Cognito 的访问权限的详细信息。要查看您可在 IAM 中使用的 Amazon Cognito 基于身份的策略示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例。

使用身份进行身份验证

身份验证是您 Amazon 使用身份凭证登录的方式。您必须以 IAM 用户身份或通过担 Amazon Web Services 账户根用户任 IAM 角色进行身份验证（登录 Amazon）。

如果您 Amazon 以编程方式访问，则会 Amazon 提供软件开发套件 (SDK) 和命令行接口 (CLI)，以便使用您的凭据对请求进行加密签名。如果您不使用 Amazon 工具，则必须自己签署请求。有关使用推荐的方法自行签署请求的更多信息，请参阅《IAM 用户指南》中的用于签署 API 请求的Amazon 签名版本 4。

无论使用何种身份验证方法，您都可能需要提供其他安全信息。例如， Amazon 建议您使用多重身份验证 (MFA) 来提高账户的安全性。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 中的Amazon 多重身份验证。

Amazon Web Services 账户 root 用户

创建时 Amazon Web Services 账户，首先要有一个登录身份，该身份可以完全访问账户中的所有资源 Amazon Web Services 服务 和资源。此身份被称为 Amazon Web Services 账户 root 用户，使用您创建账户时使用的电子邮件地址和密码登录即可访问该身份。强烈建议您不要使用根用户执行日常任务。保护好根用户凭证，并使用这些凭证来执行仅根用户可以执行的任务。有关要求您以根用户身份登录的任务的完整列表，请参阅 IAM 用户指南中的需要根用户凭证的任务。

联合身份

作为最佳实践，要求人类用户（包括需要管理员访问权限的用户）使用与身份提供商的联合身份验证 Amazon Web Services 服务 通过临时证书进行访问。

联合身份是指您的企业用户目录、Web 身份提供商、Identity C 或者任何使用 Amazon Web Services 服务 通过身份源提供的凭据进行访问的用户。 Amazon Directory Service当联合身份访问时 Amazon Web Services 账户，他们将扮演角色，角色提供临时证书。

IAM 用户和群组

I AM 用户是您 Amazon Web Services 账户 内部对个人或应用程序具有特定权限的身份。在可能的情况下，我们建议使用临时凭证，而不是创建具有长期凭证（如密码和访问密钥）的 IAM 用户。但是，如果您有一些特定的使用场景需要长期凭证以及 IAM 用户，建议您轮换访问密钥。有关更多信息，请参阅《IAM 用户指南》中的对于需要长期凭证的用例，应在需要时更新访问密钥。

IAM 组是一个指定一组 IAM 用户的身份。您不能使用组的身份登录。您可以使用组来一次性为多个用户指定权限。如果有大量用户，使用组可以更轻松地管理用户权限。例如，您可以拥有一个名为的群组，IAMAdmins并向该群组授予管理 IAM 资源的权限。

用户与角色不同。用户唯一地与某个人员或应用程序关联，而角色旨在让需要它的任何人代入。用户具有永久的长期凭证，而角色提供临时凭证。要了解更多信息，请参阅《IAM 用户指南》中的 IAM 用户的使用案例。

IAM 角色

I AM 角色是您内部具有特定权限 Amazon Web Services 账户 的身份。它类似于 IAM 用户，但与特定人员不关联。要在中临时担任 IAM 角色 Amazon Web Services Management Console，您可以从用户切换到 IAM 角色（控制台）。您可以通过调用 Amazon CLI 或 Amazon API 操作或使用自定义 URL 来代入角色。有关使用角色的方法的更多信息，请参阅《IAM 用户指南》中的代入角色的方法。

具有临时凭证的 IAM 角色在以下情况下很有用：

- 联合用户访问：要向联合身份分配权限，请创建角色并为角色定义权限。当联合身份进行身份验证时，该身份将与角色相关联并被授予由此角色定义的权限。有关用于联合身份验证的角色的信息，请参阅《IAM 用户指南》中的针对第三方身份提供商创建角色（联合身份验证）。
- 临时 IAM 用户权限：IAM 用户可代入 IAM 用户或角色，以暂时获得针对特定任务的不同权限。
- 跨账户存取：您可以使用 IAM 角色以允许不同账户中的某个人（可信主体）访问您的账户中的资源。角色是授予跨账户访问权限的主要方式。但是，对于某些资源 Amazon Web Services 服务，您可以将策略直接附加到资源（而不是使用角色作为代理）。要了解用于跨账户访问的角色和基于资源的策略之间的差别，请参阅 IAM 用户指南中的 IAM 中的跨账户资源访问。
- 跨服务访问 — 有些 Amazon Web Services 服务 使用其他 Amazon Web Services 服务服务中的功能。例如，当您在服务中拨打电话时，该服务通常会在 Amazon 中运行应用程序 EC2 或在 Amazon

S3 中存储对象。服务可能会使用发出调用的主体的权限、使用服务角色或使用服务相关角色来执行此操作。

- 转发访问会话 (FAS) — 当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 Amazon Web Services 服务 向下游服务发出请求的请求。 Amazon Web Services 服务只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。
- 服务角色 - 服务角色是服务代表您在您的账户中执行操作而分派的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。
- 服务相关角色-服务相关角色是一种链接到的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

- 在 A@@ mazon 上运行的应用程序 EC2 — 您可以使用 IAM 角色管理在 EC2 实例上运行并发出 Amazon CLI 或 Amazon API 请求的应用程序的临时证书。这比在 EC2 实例中存储访问密钥更可取。要为 EC2 实例分配 Amazon 角色并使其可供其所有应用程序使用，您需要创建一个附加到该实例的实例配置文件。实例配置文件包含该角色，并允许在 EC2 实例上运行的程序获得临时证书。有关更多信息，请参阅 IAM 用户指南中的使用 IAM 角色向在 A mazon EC2 实例上运行的应用程序授予权限。

使用策略管理访问

您可以 Amazon 通过创建策略并将其附加到 Amazon 身份或资源来控制中的访问权限。策略是其中的一个对象 Amazon ，当与身份或资源关联时，它会定义其权限。 Amazon 在委托人（用户、root 用户或角色会话）发出请求时评估这些策略。策略中的权限确定是允许还是拒绝请求。大多数策略都以 JSON 文档的 Amazon 形式存储在中。有关 JSON 策略文档的结构和内容的更多信息，请参阅 IAM 用户指南中的 JSON 策略概览。

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

默认情况下，用户和角色没有权限。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

IAM 策略定义操作的权限，无关乎您使用哪种方法执行操作。例如，假设您有一个允许 `iam:GetRole` 操作的策略。拥有该策略的用户可以从 Amazon Web Services Management Console Amazon CLI、或 Amazon API 获取角色信息。

基于身份的策略

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户托管策略定义自定义 IAM 权限。

基于身份的策略可以进一步归类为内联策略或托管式策略。内联策略直接嵌入单个用户、组或角色中。托管策略是独立的策略，您可以将其附加到中的多个用户、群组和角色 Amazon Web Services 账户。托管策略包括 Amazon 托管策略和客户托管策略。要了解如何在托管策略和内联策略之间进行选择，请参阅《IAM 用户指南》中的在托管策略与内联策略之间进行选择。

基于资源的策略

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

基于资源的策略是位于该服务中的内联策略。您不能在基于资源的策略中使用 IAM 中的 Amazon 托管策略。

访问控制列表 (ACLs)

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

Amazon S3 和 Amazon VPC 就是支持的服务示例 ACLs。 Amazon WAF要了解更多信息 ACLs，请参阅《亚马逊简单存储服务开发者指南》中的访问控制列表 (ACL) 概述。

其他策略类型

Amazon 支持其他不太常见的策略类型。这些策略类型可以设置更常用的策略类型向您授予的最大权限。

- 权限边界：权限边界是一个高级特征，用于设置基于身份的策略可以为 IAM 实体（IAM 用户或角色）授予的最大权限。您可为实体设置权限边界。这些结果权限是实体基于身份的策略及其权限边界

的交集。在 `Principal` 中指定用户或角色的基于资源的策略不受权限边界限制。任一项策略中的显式拒绝将覆盖允许。有关权限边界的更多信息，请参阅IAM 用户指南中的 IAM 实体的权限边界。

- 服务控制策略 (SCPs)- SCPs 是指定组织或组织单位 (OU) 的最大权限的 JSON 策略 Amazon Organizations。 Amazon Organizations 是一项用于对您的企业拥有的多 Amazon Web Services 账户 项进行分组和集中管理的服务。如果您启用组织中的所有功能，则可以将服务控制策略 (SCPs) 应用于您的任何或所有账户。SCP 限制成员账户中的实体（包括每个 Amazon Web Services 账户根用户实体）的权限。有关 Organization SCPs s 和的更多信息，请参阅《Amazon Organizations 用户指南》中的服务控制策略。
- 资源控制策略 (RCPs) — RCPs 是 JSON 策略，您可以使用它来设置账户中资源的最大可用权限，而无需更新附加到您拥有的每个资源的 IAM 策略。RCP 限制成员账户中资源的权限，并可能影响身份（包括身份）的有效权限 Amazon Web Services 账户根用户，无论这些身份是否属于您的组织。有关 Organizations 的更多信息 RCPs，包括 Amazon Web Services 服务 该支持的列表 RCPs，请参阅《Amazon Organizations 用户指南》中的资源控制策略 (RCPs)。
- 会话策略：会话策略是当您以编程方式为角色或联合用户创建临时会话时作为参数传递的高级策略。结果会话的权限是用户或角色的基于身份的策略和会话策略的交集。权限也可以来自基于资源的策略。任一项策略中的显式拒绝将覆盖允许。有关更多信息，请参阅 IAM 用户指南中的会话策略。

多个策略类型

当多个类型的策略应用于一个请求时，生成的权限更加复杂和难以理解。要了解在涉及多种策略类型时如何 Amazon 确定是否允许请求，请参阅 IAM 用户指南中的策略评估逻辑。

Amazon Cognito 如何与 IAM 配合使用

在使用 IAM 管理对 Amazon Cognito 的访问权限之前，您应该了解哪些 IAM 功能可用于 Amazon Cognito。

可与 Amazon Cognito 一起使用的 IAM 功能

IAM 特征	Amazon Cognito 支持
基于身份的策略	是
基于资源的策略	否
策略操作	是

IAM 特征	Amazon Cognito 支持
策略资源	是
策略条件键	是
ACLs	否
ABAC（策略中的标签）	部分
临时凭证	是
主体权限	否
服务角色	是
服务相关角色	是

要全面了解 Amazon Cognito 和其他 Amazon 服务如何与大多数 IAM 功能配合使用，请参阅 IAM 用户指南中与 IAM 配合使用的Amazon 服务。

Amazon Cognito 基于身份的策略

支持基于身份的策略：是

基于身份的策略是可附加到身份（如 IAM 用户、用户组或角色）的 JSON 权限策略文档。这些策略控制用户和角色可在何种条件下对哪些资源执行哪些操作。要了解如何创建基于身份的策略，请参阅《IAM 用户指南》中的使用客户管理型策略定义自定义 IAM 权限。

通过使用 IAM 基于身份的策略，您可以指定允许或拒绝的操作和资源以及允许或拒绝操作的条件。您无法在基于身份的策略中指定主体，因为它适用于其附加的用户或角色。要了解可在 JSON 策略中使用的所有元素，请参阅《IAM 用户指南》中的 IAM JSON 策略元素引用。

适用于 Amazon Cognito 的基于身份的策略示例

要查看 Amazon Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例。

Amazon Cognito 内基于资源的策略

支持基于资源的策略：否

基于资源的策略是附加到资源的 JSON 策略文档。基于资源的策略的示例包括 IAM 角色信任策略和 Amazon S3 存储桶策略。在支持基于资源的策略的服务中，服务管理员可以使用它们来控制对特定资源的访问。对于在其中附加策略的资源，策略定义指定主体可以对该资源执行哪些操作以及在什么条件下执行。您必须在基于资源的策略中指定主体。委托人可以包括账户、用户、角色、联合用户或 Amazon Web Services 服务。

要启用跨账户访问，您可以将整个账户或其他账户中的 IAM 实体指定为基于资源的策略中的主体。将跨账户主体添加到基于资源的策略只是建立信任关系工作的一半而已。当委托人和资源处于不同位置时 Amazon Web Services 账户，可信账户中的 IAM 管理员还必须向委托人实体（用户或角色）授予访问资源的权限。他们通过将基于身份的策略附加到实体以授予权限。但是，如果基于资源的策略向同一个账户中的主体授予访问权限，则不需要额外的基于身份的策略。有关更多信息，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

Amazon Cognito 的策略操作

支持策略操作：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

JSON 策略的 `Action` 元素描述可用于在策略中允许或拒绝访问的操作。策略操作通常与关联的 Amazon API 操作同名。有一些例外情况，例如没有匹配 API 操作的仅限权限 操作。还有一些操作需要在策略中执行多个操作。这些附加操作称为相关操作。

在策略中包含操作以授予执行关联操作的权限。

要查看 Amazon Cognito 操作的列表，请参阅《服务授权参考》中的 Amazon Cognito 定义的操作。

Amazon Cognito 中的策略操作在操作前使用以下前缀：

```
cognito-identity
```

要在单个语句中指定多项操作，请使用逗号将它们隔开。

```
"Action": [
      "cognito-identity:action1",
      "cognito-identity:action2"
         ]
```

已签名与未签名 APIs

当您使用 Amazon 证书签署 Amazon Cognito API 请求时，您可以在 Amazon Identity and Access Management (IAM) 策略中对其进行限制。您必须使用 Amazon 凭证签署的 API 请求包括使用 `AdminInitiateAuth` 进行服务器端登录，以及创建、查看或修改您的 Amazon Cognito 资源的操作（如 `UpdateUserPool`）。有关已签名的 API 请求的更多信息，请参阅签署 Amazon API 请求。

由于 Amazon Cognito 是一款用于您想要向公众开放的应用程序的消费者身份产品，因此您可以访问以下未签名的应用程序。 APIs您的应用程序针对您的用户和潜在用户发出这些 API 请求。有些 APIs 不需要事先授权`InitiateAuth`，例如启动新的身份验证会话。有些人 APIs 使用访问令牌或会话密钥进行授权，例如`VerifySoftwareToken`为现有经过身份验证的会话的用户完成 MFA 设置。未签署但已授权的 Amazon Cognito 用户池 API 支持请求语法中的 `Session` 或 `AccessToken` 参数，如 Amazon Cognito API 参考中所示。未签署的 Amazon Cognito 身份 API 支持 `IdentityId` 参数，如 Amazon Cognito 联合身份 API 参考中所示。

有关 Amazon Cognito 用户池 API 操作的授权模式和角色的更多信息，请参阅按授权模型分组的 API 操作列表。

Amazon Cognito 身份池 API 操作

- `GetId`
- `GetOpenIdToken`
- `GetCredentialsForIdentity`
- `UnlinkIdentity`

Amazon Cognito 用户池 API 操作

- `AssociateSoftwareToken`
- `ChangePassword`
- `ConfirmDevice`
- `ConfirmForgotPassword`
- `ConfirmSignUp`
- `DeleteUser`
- `DeleteUserAttributes`
- `ForgetDevice`
- `ForgotPassword`

- GetDevice
- GetUser
- GetUserAttributeVerificationCode
- GlobalSignOut
- InitiateAuth
- ListDevices
- ResendConfirmationCode
- RespondToAuthChallenge
- RevokeToken
- SetUserMFAPreference
- SetUserSettings
- SignUp
- UpdateAuthEventFeedback
- UpdateDeviceStatus
- UpdateUserAttributes
- VerifySoftwareToken
- VerifyUserAttribute

要查看 Amazon Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例。

Amazon Cognito 的策略资源

支持策略资源：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

Resource JSON 策略元素指定要向其应用操作的一个或多个对象。语句必须包含 Resource 或 NotResource 元素。作为最佳实践，请使用其 Amazon 资源名称（ARN）指定资源。对于支持特定资源类型（称为资源级权限）的操作，您可以执行此操作。

对于不支持资源级权限的操作（如列出操作），请使用通配符（*）指示语句应用于所有资源。

```
"Resource": "*"
```

亚马逊资源名称 (ARNs)

ARNs 适用于 Amazon Cognito 联合身份

在 Amazon Cognito 身份池（联合身份）中，您可以使用如下例所示的 Amazon Resource Name（ARN）格式限制 IAM 用户对特定身份池的访问权限。有关更多信息 ARNs，请参阅 IAM 标识符。

```
arn:aws:cognito-identity:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

ARNs 适用于亚马逊 Cognito Sync

在 Amazon Cognito Sync 中，客户还可以按身份池 ID、身份 ID 和数据集名称限制访问。

对于在身份池上进行操作 APIs ，身份池 ARN 格式与 Amazon Cognito Federations 的格式相同，唯一的不同是服务名称`cognito-sync`取代了：`cognito-identity`

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID
```

为 APIs 此，可以对单个身份进行操作，例如`RegisterDevice`，您可以通过以下 ARN 格式引用个人身份：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID
```

为 APIs 此，可以对数据集（例如`UpdateRecords`和）进行操作`ListRecords`，您可以使用以下 ARN 格式引用单个数据集：

```
arn:aws:cognito-sync:REGION:ACCOUNT_ID:identitypool/IDENTITY_POOL_ID/
identity/IDENTITY_ID/dataset/DATASET_NAME
```

ARNs 适用于 Amazon Cognito 用户池

对于 Amazon Cognito 中的您的用户池，您可以使用以下 ARN 格式限制用户对特定用户池的访问权限：

```
arn:aws:cognito-idp:REGION:ACCOUNT_ID:userpool/USER_POOL_ID
```

要查看 Amazon Cognito 资源类型及其列表 ARNs，请参阅《服务授权参考》中的 Amazon Cognito 定义的资源。要了解您可以使用哪些操作指定每个资源的 ARN，请参阅 Amazon Cognito 定义的操作。

要查看 Amazon Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例。

Amazon Cognito 的策略条件键

支持特定于服务的策略条件键：是

管理员可以使用 Amazon JSON 策略来指定谁有权访问什么。也就是说，哪个主体可以对什么资源执行操作，以及在什么条件下执行。

在 `Condition` 元素（或 `Condition` 块）中，可以指定语句生效的条件。`Condition` 元素是可选的。您可以创建使用条件运算符（例如，等于或小于）的条件表达式，以使策略中的条件与请求中的值相匹配。

如果您在一个语句中指定多个 `Condition` 元素，或在单个 `Condition` 元素中指定多个键，则 Amazon 使用逻辑 AND 运算评估它们。如果您为单个条件键指定多个值，则使用逻辑OR运算来 Amazon 评估条件。在授予语句的权限之前必须满足所有的条件。

在指定条件时，您也可以使用占位符变量。例如，只有在使用 IAM 用户名标记 IAM 用户时，您才能为其授予访问资源的权限。有关更多信息，请参阅《IAM 用户指南》中的 IAM 策略元素：变量和标签。

Amazon 支持全局条件密钥和特定于服务的条件密钥。要查看所有 Amazon 全局条件键，请参阅 IAM 用户指南中的Amazon 全局条件上下文密钥。

有关 Amazon Cognito 条件键的列表，请参阅《服务授权参考》中的 Amazon Cognito 的条件键。要了解您可以对哪些操作和资源使用条件键，请参阅 Amazon Cognito 定义的操作。

要查看 Amazon Cognito 基于身份的策略的示例，请参阅适用于 Amazon Cognito 的基于身份的策略示例。

亚马逊 Cognito 中的访问控制列表 (ACLs)

支持 ACLs：否

访问控制列表 (ACLs) 控制哪些委托人（账户成员、用户或角色）有权访问资源。 ACLs 与基于资源的策略类似，尽管它们不使用 JSON 策略文档格式。

使用 Amazon Cognito 的基于属性的访问权限控制（ABAC）

支持 ABAC（策略中的标签）：部分支持

基于属性的访问控制（ABAC）是一种授权策略，该策略基于属性来定义权限。在中 Amazon，这些属性称为标签。您可以向 IAM 实体（用户或角色）和许多 Amazon 资源附加标签。标记实体和资源是 ABAC 的第一步。然后设计 ABAC 策略，以在主体的标签与他们尝试访问的资源标签匹配时允许操作。

ABAC 在快速增长的环境中非常有用，并在策略管理变得繁琐的情况下可以提供帮助。

要基于标签控制访问，您需要使用 `aws:ResourceTag/key-name`、`aws:RequestTag/key-name` 或 `aws:TagKeys` 条件键在策略的条件元素中提供标签信息。

如果某个服务对于每种资源类型都支持所有这三个条件键，则对于该服务，该值为是。如果某个服务仅对于部分资源类型支持所有这三个条件键，则该值为部分。

有关 ABAC 的更多信息，请参阅《IAM 用户指南》中的使用 ABAC 授权定义权限。要查看设置 ABAC 步骤的教程，请参阅《IAM 用户指南》中的使用基于属性的访问权限控制（ABAC）。

将临时凭证用于 Amazon Cognito

支持临时凭证：是

当你使用临时证书登录时，有些 Amazon Web Services 服务 不起作用。有关更多信息，包括哪些 Amazon Web Services 服务 适用于临时证书，请参阅 IAM 用户指南中的Amazon Web Services 服务与 IA M 配合使用的信息。

如果您使用除用户名和密码之外的任何方法登录，则 Amazon Web Services Management Console 使用的是临时证书。例如，当您 Amazon 使用公司的单点登录 (SSO) 链接进行访问时，该过程会自动创建临时证书。当您以用户身份登录控制台，然后切换角色时，您还会自动创建临时凭证。有关切换角色的更多信息，请参阅《IAM 用户指南》中的从用户切换到 IAM 角色（控制台）。

您可以使用 Amazon CLI 或 Amazon API 手动创建临时证书。然后，您可以使用这些临时证书进行访问 Amazon。 Amazon 建议您动态生成临时证书，而不是使用长期访问密钥。有关更多信息，请参阅 IAM 中的临时安全凭证。

Amazon Cognito 的跨服务主体权限

支持转发访问会话（FAS）：否

当您使用 IAM 用户或角色在中执行操作时 Amazon，您被视为委托人。使用某些服务时，您可能会执行一个操作，然后此操作在其他服务中启动另一个操作。FAS 使用调用委托人的权限以及 Amazon Web Services 服务 向下游服务发出请求的请求。 Amazon Web Services 服务只有当服务收到需要与其他 Amazon Web Services 服务 或资源交互才能完成的请求时，才会发出 FAS 请求。在这种情况下，您必须具有执行这两项操作的权限。有关发出 FAS 请求时的策略详情，请参阅转发访问会话。

Amazon Cognito 的服务角色

支持服务角色：是

服务角色是由一项服务担任、代表您执行操作的 IAM 角色。IAM 管理员可以在 IAM 中创建、修改和删除服务角色。有关更多信息，请参阅《IAM 用户指南》中的创建向 Amazon Web Services 服务委派权限的角色。

有关 Amazon Cognito 服务角色的详细信息，请参阅激活推送同步和实施推送同步。

> **Warning**
> 更改服务角色的权限可能会破坏 Amazon Cognito 的功能。仅当 Amazon Cognito 提供相关指导时才编辑服务角色。

Amazon Cognito 的服务相关角色

支持服务相关角色：是

服务相关角色是一种链接到的服务角色。 Amazon Web Services 服务服务可以代入代表您执行操作的角色。服务相关角色出现在您的中 Amazon Web Services 账户 ，并且归服务所有。IAM 管理员可以查看但不能编辑服务相关角色的权限。

有关创建或管理 Amazon Cognito 服务相关角色的详细信息，请参阅对 Amazon Cognito 使用服务相关角色。

适用于 Amazon Cognito 的基于身份的策略示例

原定设置情况下，用户和角色没有创建或修改 Amazon Cognito 资源的权限。他们也无法使用 Amazon Web Services Management Console、 Amazon Command Line Interface (Amazon CLI) 或 Amazon API 执行任务。要授予用户对所需资源执行操作的权限，IAM 管理员可以创建 IAM 策略。管理员随后可以向角色添加 IAM 策略，用户可以代入角色。

要了解如何使用这些示例 JSON 策略文档创建基于 IAM 身份的策略，请参阅《IAM 用户指南》中的创建 IAM 策略（控制台）。

有关 Amazon Cognito 定义的操作和资源类型（包括每种资源类型的格式）的详细信息，请参阅《服务授权参考》中的 Amazon Cognito 的操作、资源和条件密钥。 ARNs

主题

- 策略最佳实践
- 使用 Amazon Cognito 控制台
- 允许用户查看他们自己的权限
- 限制对特定身份池的控制台访问权限
- 允许池中的所有身份访问特定数据集

策略最佳实践

基于身份的策略确定某个人是否可以创建、访问或删除您账户中的 Amazon Cognito 资源。这些操作可能会使 Amazon Web Services 账户产生成本。创建或编辑基于身份的策略时，请遵循以下指南和建议：

- 开始使用 Amazon 托管策略并转向最低权限权限 — 要开始向用户和工作负载授予权限，请使用为许多常见用例授予权限的Amazon 托管策略。它们在你的版本中可用 Amazon Web Services 账户。我们建议您通过定义针对您的用例的 Amazon 客户托管策略来进一步减少权限。有关更多信息，请参阅《IAM 用户指南》中的 Amazon 托管式策略或工作职能的Amazon 托管式策略。
- 应用最低权限：在使用 IAM 策略设置权限时，请仅授予执行任务所需的权限。为此，您可以定义在特定条件下可以对特定资源执行的操作，也称为最低权限许可。有关使用 IAM 应用权限的更多信息，请参阅《IAM 用户指南》中的 IAM 中的策略和权限。
- 使用 IAM 策略中的条件进一步限制访问权限：您可以向策略添加条件来限制对操作和资源的访问。例如，您可以编写策略条件来指定必须使用 SSL 发送所有请求。如果服务操作是通过特定的方式使用的，则也可以使用条件来授予对服务操作的访问权限 Amazon Web Services 服务，例如 Amazon CloudFormation。有关更多信息，请参阅《IAM 用户指南》中的 IAM JSON 策略元素：条件。
- 使用 IAM Access Analyzer 验证您的 IAM 策略，以确保权限的安全性和功能性 – IAM Access Analyzer 会验证新策略和现有策略，以确保策略符合 IAM 策略语言（JSON）和 IAM 最佳实践。IAM Access Analyzer 提供 100 多项策略检查和可操作的建议，以帮助您制定安全且功能性强的策略。有关更多信息，请参阅《IAM 用户指南》中的使用 IAM Access Analyzer 验证策略。

- 需要多重身份验证 (MFA)-如果 Amazon Web Services 账户您的场景需要 IAM 用户或根用户，请启用 MFA 以提高安全性。若要在调用 API 操作时需要 MFA，请将 MFA 条件添加到您的策略中。有关更多信息，请参阅《IAM 用户指南》中的使用 MFA 保护 API 访问。

有关 IAM 中的最佳实操的更多信息，请参阅《IAM 用户指南》中的 IAM 中的安全最佳实践。

> **Note**
>
> 当您查看和修改 Amazon Cognito 资源时，Amazon Cognito 控制台的原始版本和新版本具有不同的底层行为。如果您仅在条件 `aws:ViaAWSService` 为 true 时授予对 `cognito-idp` 服务前缀下的操作的权限，受影响的 IAM 主体可能在原始控制台中对 Amazon Cognito 资源有效，但在 Amazon Cognito 控制台中无效。要在 Amazon Cognito 控制台中有效，请不要在您的 IAM policy 中对 Amazon Cognito 权限设置 `aws:ViaAWSService` 条件。

使用 Amazon Cognito 控制台

要访问 Amazon Cognito 控制台，您必须具有一组最低的权限。这些权限必须允许您列出和查看有关您的 Amazon Cognito 资源的详细信息。 Amazon Web Services 账户如果创建比必需的最低权限更为严格的基于身份的策略，对于附加了该策略的实体（用户或角色），控制台将无法按预期正常运行。

对于仅调用 Amazon CLI 或 Amazon API 的用户，您无需为其设置最低控制台权限。相反，只允许访问与其尝试执行的 API 操作相匹配的操作。

为确保用户和角色仍然可以使用 Amazon Cognito 控制台，还需要将亚马逊 Co `ConsoleAccess` gnito `ReadOnly` Amazon 或托管策略附加到实体。有关更多信息，请参阅《IAM 用户指南》中的为用户添加权限。

允许用户查看他们自己的权限

该示例说明了您如何创建策略，以允许 IAM 用户查看附加到其用户身份的内联和托管式策略。此策略包括在控制台上或使用 Amazon CLI 或 Amazon API 以编程方式完成此操作的权限。

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ViewOwnUserInfo",
            "Effect": "Allow",
```

```
            "Action": [
                "iam:GetUserPolicy",
                "iam:ListGroupsForUser",
                "iam:ListAttachedUserPolicies",
                "iam:ListUserPolicies",
                "iam:GetUser"
            ],
            "Resource": ["arn:aws:iam::*:user/${aws:username}"]
        },
        {
            "Sid": "NavigateInConsole",
            "Effect": "Allow",
            "Action": [
                "iam:GetGroupPolicy",
                "iam:GetPolicyVersion",
                "iam:GetPolicy",
                "iam:ListAttachedGroupPolicies",
                "iam:ListGroupPolicies",
                "iam:ListPolicyVersions",
                "iam:ListPolicies",
                "iam:ListUsers"
            ],
            "Resource": "*"
        }
    ]
}
```

限制对特定身份池的控制台访问权限

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:ListIdentityPools"
      ],
      "Resource": "*"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-identity:*"
```

```
      ],
      "Resource": "arn:aws:cognito-identity:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    },
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:*"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678"
    }
  ]
}
```

允许池中的所有身份访问特定数据集

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "cognito-sync:ListRecords",
        "cognito-sync:UpdateRecords"
      ],
      "Resource": "arn:aws:cognito-sync:us-east-1:0123456789:identitypool/us-
east-1:1a1a1a1a-ffff-1111-9999-12345678/identity/*/dataset/UserProfile"
    }
  ]
}
```

Amazon Cognito 身份和访问问题排查

使用以下信息帮助您诊断和修复在使用 Amazon Cognito 和 IAM 时可能遇到的常见问题。

主题

- 我没有在 Amazon Cognito 中执行操作的权限
- 我无权执行 iam：PassRole

- 我是管理员并希望允许其他人访问 Amazon Cognito
- 我想允许 Amazon 账户以外的人访问我的 Amazon Cognito 资源

我没有在 Amazon Cognito 中执行操作的权限

如果您收到错误提示，指明您无权执行某个操作，则必须更新策略以允许执行该操作。

当 `mateojackson` IAM 用户尝试使用控制台查看有关虚构 *`my-example-widget`* 资源的详细信息，但不拥有虚构 `cognito-identity:`*`GetWidget`* 权限时，会发生以下示例错误。

```
User: arn:aws:iam::123456789012:user/mateojackson is not authorized to perform:
 cognito-identity:GetWidget on resource: my-example-widget
```

在此情况下，必须更新 `mateojackson` 用户的策略，以允许使用 `cognito-identity:`*`GetWidget`* 操作访问 *`my-example-widget`* 资源。

如果您需要帮助，请联系您的 Amazon 管理员。您的管理员是提供登录凭证的人。

我无权执行 iam：PassRole

如果您收到一个错误，表明您无权执行 `iam:PassRole` 操作，则必须更新策略以允许您将角色传递给 Amazon Cognito。

有些 Amazon Web Services 服务 允许您将现有角色传递给该服务，而不是创建新的服务角色或服务相关角色。为此，您必须具有将角色传递到服务的权限。

当名为 `marymajor` 的 IAM 用户尝试使用控制台在 Amazon Cognito 中执行操作时，会发生以下示例错误。但是，服务必须具有服务角色所授予的权限才可执行此操作。Mary 不具有将角色传递到服务的权限。

```
User: arn:aws:iam::123456789012:user/marymajor is not authorized to perform:
 iam:PassRole
```

在这种情况下，必须更新 Mary 的策略以允许她执行 `iam:PassRole` 操作。

如果您需要帮助，请联系您的 Amazon 管理员。您的管理员是提供登录凭证的人。

我是管理员并希望允许其他人访问 Amazon Cognito

要允许其他人访问 Amazon Cognito，您必须向需要访问的人员或应用程序授予权限。如果使用 Amazon IAM Identity Center 管理人员和应用程序，则可以向用户或组分配权限集来定义其访问权限级

别。权限集会自动创建 IAM 策略并将其分配给与人员或应用程序关联的 IAM 角色。有关更多信息，请参阅《Amazon IAM Identity Center 用户指南》中的权限集。

如果未使用 IAM Identity Center，则必须为需要访问的人员或应用程序创建 IAM 实体（用户或角色）。然后，您必须将策略附加到实体，以便在 Amazon Cognito 中向其授予正确的权限。授予权限后，向用户或应用程序开发人员提供凭证。他们将使用这些凭证访问 Amazon。要了解有关创建 IAM 用户、组、策略和权限的更多信息，请参阅《IAM 用户指南》中的 IAM 身份和 IAM 中的策略和权限。

我想允许 Amazon 账户以外的人访问我的 Amazon Cognito 资源

您可以创建一个角色，以便其他账户中的用户或您组织外的人员可以使用该角色来访问您的资源。您可以指定谁值得信赖，可以代入角色。对于支持基于资源的策略或访问控制列表 (ACLs) 的服务，您可以使用这些策略向人们授予访问您的资源的权限。

要了解更多信息，请参阅以下内容：

- 要了解 Amazon Cognito 是否支持这些功能，请参阅Amazon Cognito 如何与 IAM 配合使用。
- 要了解如何提供对您拥有的资源的访问权限 Amazon Web Services 账户 ，请参阅 IAM 用户指南中的向您拥有 Amazon Web Services 账户 的另一个 IAM 用户提供访问权限。
- 要了解如何向第三方提供对您的资源的访问权限 Amazon Web Services 账户，请参阅 IAM 用户指南中的向第三方提供访问权限。 Amazon Web Services 账户
- 要了解如何通过身份联合验证提供访问权限，请参阅《IAM 用户指南》中的为经过外部身份验证的用户（身份联合验证）提供访问权限。
- 要了解使用角色和基于资源的策略进行跨账户访问之间的差别，请参阅《IAM 用户指南》中的 IAM 中的跨账户资源访问。

对 Amazon Cognito 使用服务相关角色

Amazon Cognito 使用 Amazon Identity and Access Management (IAM) 服务相关角色。服务相关角色是一种独特的 IAM 角色，其信任策略 Amazon Web Services 服务 允许担任该角色。服务相关角色由 Amazon Cognito 预定义，包括该服务代表您调用 Amazon 其他服务所需的所有权限。

服务相关角色可让您更轻松地设置 Amazon Cognito，因为您不必手动添加必要的权限。Amazon Cognito 定义其服务相关角色的权限，除非另外定义，否则只有 Amazon Cognito 可以代入该角色。定义的权限包括信任策略和权限策略，以及不能附加到任何其他 IAM 实体的权限策略。

只有在首先删除相关资源后，您才能删除服务相关角色。这将保护您的 Amazon Cognito 资源，因为您不会无意中删除对资源的访问权限。

有关支持服务相关角色的其它服务的信息，请参阅使用 IAM 的Amazon 服务并查找服务相关角色列中显示为是的服务。选择是和链接，查看该服务的服务相关角色文档。

Amazon Cognito 的服务相关角色权限

Amazon Cognito 使用下列服务相关角色：

- AWSServiceRoleForAmazonCognitoIdpEmailService— 允许 Amazon Cognito 用户池服务使用你的 Amazon SES 身份发送电子邮件。

- AWSServiceRoleForAmazonCognitoIdp— 允许 Amazon Cognito 用户池为你的 Amazon Pinpoint 项目发布事件和配置终端节点。

AWSServiceRoleForAmazonCognitoIdpEmailService

`AWSServiceRoleForAmazonCognitoIdpEmailService` 服务相关角色信任以下服务代入该角色：

- `email.cognito-idp.amazonaws.com`

角色权限策略允许 Amazon Cognito 对指定资源完成以下操作：

允许的操作 AWSServiceRoleForAmazonCognitoIdpEmailService：

- 操作：`ses:SendEmail` 和 `ses:SendRawEmail`
- 资源：*

此策略拒绝 Amazon Cognito 对指定资源完成以下操作的功能：

拒绝的操作

- 操作：`ses:List*`
- 资源：*

凭借这些权限，Amazon Cognito 只能使用 Amazon SES 中经过验证的电子邮件地址向用户发送电子邮件。当您的用户在客户端应用程序中针对用户池执行特定操作（如注册或重置密码）时，Amazon Cognito 将向用户发送电子邮件。

您必须配置权限，允许 IAM 实体（如用户、组或角色）创建、编辑或删除服务相关角色。有关更多信息，请参阅《IAM 用户指南》中的服务相关角色权限。

AWSServiceRoleForAmazonCognitoIdp

AWSServiceRoleForAmazonCognitoIdp 服务相关角色信任以下服务来代入该角色：

- `email.cognito-idp.amazonaws.com`

角色权限策略允许 Amazon Cognito 对指定资源完成以下操作：

允许的操作 AWSService RoleForAmazonCognitoIdp

- 操作：`cognito-idp:Describe`
- 资源：`*`

有了此权限，Amazon Cognito 可以为您调用 `Describe` Amazon Cognito API 操作。

> **Note**
>
> 当您将 Amazon Cognito 与采用 `createUserPoolClient` 和 `updateUserPoolClient` 的 Amazon Pinpoint 集成时，资源权限将作为内联策略添加到 SLR 中。内联策略将提供 `mobiletargeting:UpdateEndpoint` 和 `mobiletargeting:PutEvents` 权限。这些权限允许 Amazon Cognito 发布事件并为与 Cognito 集成的 Pinpoint 项目配置端点。

创建适用于 Amazon Cognito 的服务相关角色

您无需手动创建服务相关角色。当您将用户池配置为使用您的 Amazon SES 配置来处理在 Amazon Web Services Management Console Amazon CLI、或 Amazon Cognito API 中发送电子邮件时，Amazon Cognito 会为您创建服务相关角色。

如果您删除该服务相关角色，然后需要再次创建，您可以使用相同流程在账户中重新创建此角色。当您将用户池配置为使用 Amazon SES 配置去处理邮件送达时，Amazon Cognito 会为您创建与服务相关的角色。

在 Amazon Cognito 可以创建此角色之前，您用来设置用户池的 IAM 权限必须包含 `iam:CreateServiceLinkedRole` 操作。有关更新 IAM 中权限的更多信息，请参阅《IAM 用户指南》中的更改 IAM 用户的权限。

编辑适用于 Amazon Cognito 的服务相关角色

您无法在中编辑 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 与服务相关的角色。 Amazon Identity and Access Management在创建服务相关角色后，您将无法更改角色的名称，因为可能有多种实体引用该角色。不过，您可以使用 IAM 编辑角色的说明。有关更多信息，请参阅 IAM 用户指南中的编辑服务相关角色。

删除适用于 Amazon Cognito 的服务相关角色

如果不再需要使用某个需要服务相关角色的特征或服务，我们建议您删除该角色。如果您删除角色，则只应保留 Amazon Cognito 主动监控或维护的实体。在删除角色 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 服务相关角色之前，必须对使用该角色的每个用户池执行以下操作之一：

- 删除该用户池。
- 更新用户池中的电子邮件设置以使用默认的电子邮件功能。默认设置不使用服务相关角色。

请记住使用该角色 Amazon Web Services 区域 的用户池在每个用户池中执行操作。

> **Note**
>
> 如果在您尝试删除资源时，Amazon Cognito 服务正在使用该角色，则删除操作可能会失败。如果发生这种情况，请等待几分钟后重试。

删除 Amazon Cognito 用户池

1. 登录 Amazon Web Services Management Console 并打开 Amazon Cognito 控制台，网址为。https://console.amazonaws.cn/cognito
2. 选择管理用户池。
3. 在您的用户池页面上，选择要删除的用户池。
4. 选择删除池。
5. 在删除用户池窗口中，键入 **delete**，然后选择删除池。

更新 Amazon Cognito 用户池以使用默认电子邮件功能

1. 登录 Amazon Web Services Management Console 并打开 Amazon Cognito 控制台，网址为。https://console.amazonaws.cn/cognito
2. 选择管理用户池。
3. 在您的用户池页面上，选择要更新的用户池。
4. 在左侧导航菜单中，选择消息自定义。
5. 在是否要通过 Amazon SES 配置发送电子邮件？下，选择否 -使用 Cognito（默认）。
6. 当您完成设置您的电子邮件账户选项时，选择保存更改。

使用 IAM 手动删除服务相关角色

使用 IAM 控制台 Amazon CLI、或 Amazon API 删除 AmazonCognitoIdp 或 AmazonCognitoIdpEmailService 与服务相关的角色。有关更多信息，请参阅 IAM 用户指南中的删除服务相关角色。

Amazon Cognito 服务相关角色支持的区域

Amazon Cognito 在所有提供服务 Amazon Web Services 区域 的地方都支持与服务相关的角色。有关更多信息，请参阅Amazon Web Services 区域 和端点。

Amazon Cognito 中的日志记录和监控

监控是维护 Amazon Cognito 和其他 Amazon 解决方案的可靠性、可用性和性能的重要组成部分。Amazon Cognito 目前支持下列 Amazon Web Services 服务 ，这样您便可以监控您的组织和组织内部的活动。

- Amazon CloudTrail — 通过使用， CloudTrail 您可以捕获来自亚马逊 Cognito 控制台的 API 调用，以及从对亚马逊 Cognito API 操作的代码调用中捕获 API 调用。例如，当用户进行身份验证时，CloudTrail 可以记录诸如请求中的 IP 地址、谁发出请求以及何时发出请求之类的详细信息。
- Amazon CloudWatch Logs — 借助 CloudWatch 日志，您可以向日志组发送精细的用户活动日志。例如，您可以查看详细的用户活动日志，以排查向用户发送电子邮件和短信时遇到的问题。
- Amazon CloudWatch Metrics — 借助 CloudWatch 指标，您可以近乎实时地监控、报告事件并在发生事件时自动采取行动。例如，您可以根据提供的指标创建 CloudWatch 控制面板来监控您的 Amazon Cognito 用户池，也可以根据提供的指标创建 CloudWatch 警报，以便在违反设定阈值时通知您。
- Amazon CloudWatch Logs Insig hts — 借助 CloudWatch Logs Insights，您可以配置 CloudWatch 为将事件发送 CloudTrail 到以监控 Amazon Cognito CloudTrail 日志文件。

主题

- 监控和管理成本
- 从 Amazon Cognito 用户池导出日志
- 跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况
- 亚马逊 Cognito 正在登录 Amazon CloudTrail

监控和管理成本

与其他任何方法一样 Amazon Web Services 服务，了解您的 Amazon Cognito 配置和使用对账 Amazon 单的影响非常重要。在将用户池部署到生产环境的准备过程中，为活动和资源消耗设置监控和安全防护措施。如果您知道去哪里查看费用以及哪些操作会产生额外费用，那么您就可以设置预防措施，避免账单中出现意外费用。

Amazon Cognito 会根据以下几个方面的使用情况来收费。

- 用户池每月活跃用户数 (MAUs)-比率因功能计划而异
- 使用 OIDC 或 SAML 联合身份验证 MAUs 登录的用户池
- 活跃的用户池应用程序客户端和使用客户端凭证授予的机器对机器（M2M）授权的请求量
- 某些类别的用户池的购买使用量超过了默认配额 APIs

此外，用户池的特征（例如，电子邮件消息、短信消息和 Lambda 触发器）可能会在依赖服务中产生费用。有关完整概述，请参阅 Amazon Cognito 定价。

查看和预测成本

产品发布和拓展新用户群体等大批量活动会增加 MAU 计数，并对成本产生影响。提前估算新增用户数量，并实时监控活动情况。您可能会发现需要通过购买额外的配额容量来应对用户数量的增长，或者需要通过额外的安全措施来控制新增用户量。

您可以在Amazon 账单与成本管理 控制台中查看和报告您的 Amazon 成本。您可以在账单和付款部分找到最近的 Amazon Cognito 费用。在账单、按服务计费下面，筛选 `Cognito` 以查看您的使用情况。有关更多信息，请参阅 Amazon Billing 用户指南中的查看您的账单。

要监控 API 请求速率，请在“服务配额”控制台中查看利用率指标。例如，客户端凭证请求显示为 ClientAuthentication 请求率。在您的账单中，这些请求与生成这些请求的应用程序客户端相关联。有了这些信息，您就可以在多租户架构中公平地将费用分摊给租户。

要获取一段时间内的 M2M 请求数量，您还可以将Amazon CloudTrail 事件发送到 CloudWatch Logs 进行分析。使用客户端凭证授予 CloudTrail `Token_POST`的事件查询您的事件。以下 CloudWatch Insights 查询返回此计数。

```
filter eventName = "Token_POST" and @message like '"grant_type":["client_credentials"]'
 | stats count(*)
```

管理 成本

Amazon Cognito 根据用户数量、特征使用情况和请求量来计费。以下是在 Amazon Cognito 中管理成本的一些技巧，

不要激活不活跃的用户

让用户活跃的典型操作是登录、注册和密码重置。有关更详尽的列表，请参阅每月活跃用户。Amazon Cognito 不会将不活跃用户的费用计入您的账单。避免执行任何会让用户处于活动状态的操作。不AdminGetUser使用 API 操作，而是使用该ListUsers操作查询用户。不要对包含非活跃用户的用户池进行大规模的管理操作测试。

链接联合用户

与本地用户相比，使用 SAML 2.0 或 OpenID Connect（OIDC）身份提供者登录的用户费用更高。您可以将这些用户链接到本地用户配置文件。关联用户可以作为本地用户登录，并享有联合用户附带的属性和访问权限。来自 SAML 或 OIDC IdPs 的用户在一个月内仅使用关联的本地账户登录，则按本地用户计费。

管理请求速率

如果您的用户池已接近配额上限，则可以考虑购买额外的容量来处理增加的请求量。您或许可以减少应用程序中的请求量。有关更多信息，请参阅 优化请求速率以避免达到配额限制。

仅在需要新令牌时才申请新令牌

使用客户端凭证授予的机器对机器（M2M）授权可以应对大量令牌请求。每个新的令牌请求都会影响您的请求速率配额和账单大小。为了优化成本，请在应用程序设计中包括令牌到期设置和令牌处理。

- 缓存访问令牌，以便当您的应用程序请求新令牌时，它会收到先前发放的令牌的缓存版本。实施此方法时，缓存代理充当了防护机制，防止应用程序在未意识到先前获取的令牌已过期的情况下请求访问令牌。缓存令牌非常适合 Lambda 函数和 Docker 容器等短期微服务。

- 在您的应用程序中实施令牌处理机制，从而考虑到令牌到期情况。不要等到先前的令牌即将到期时才请求新令牌。更好的做法是，在令牌生命周期达到大约 75% 时刷新令牌。这种做法可以更大限度地延长令牌持续时间，同时确保应用程序中的用户连续性。

  评估每个应用程序的机密性和可用性需求，并将用户池应用程序客户端配置为发放具有适当有效期的访问令牌。自定义令牌持续时间最适合使用寿命较长的服务器 APIs 以及能够持续管理凭证请求频率的服务器。

ListUsers，不是 AdminGetUser

要查询用户池中用户的属性，请尽可能使用 ListUsersAPI 操作和关联的 SDK 方法。 AdminGetUser将用户标记为当月活跃用户，并向用于计算用户池账单的每月活跃用户 (MAUs) 做出贡献。

删除未使用的客户端凭证应用程序客户端

M2M 授权账单基于两个因素：令牌请求速率和执行客户端凭证授予的应用程序客户端数量。当用于 M2M 授权的应用程序客户端不再使用时，请将其删除或取消它们发放客户端凭证的授权。有关管理应用程序客户端配置的更多信息，请参阅特定于应用程序的应用程序客户端设置。

管理功能计划

当您在用户池中选择功能计划时，计费率适用于用户池 MAUs 中的所有人。如果您的用户不需要更高级别的功能计划附带的功能，请将他们分成另一个用户池。

从 Amazon Cognito 用户池导出日志

您可以将用户池配置为将某些其他活动的详细日志发送给其他人（例如 CloudWatch 日志组）。Amazon Web Services 服务这些日志比中的日志更精细 Amazon CloudTrail，可用于对用户池进行故障排除和使用高级安全功能分析用户登录活动。当您想要流式传输短信和电子邮件通知错误的日志时，您的用户池会向 CloudWatch 日志组发送`ERROR`级别日志。当您想要流式传输用户登录活动的日志时，您的用户池会将 `INFO` 级别日志发送到日志组、Amazon Data Firehose 流或 Amazon S3 存储桶。您可以将这两个选项合并到一个用户池中。

主题

- 有关日志导出的需知信息
- 导出电子邮件和短信消息传输错误
- 导出威胁防护用户活动日志

有关日志导出的需知信息

成本影响

Amazon Data Firehose、Amazon S3 和 CloudWatch 日志会产生数据摄取和检索费用。您的日志配置可能会影响您的 Amazon 账单。有关更多信息，请参阅下列内容：

- Amazon CloudWatch 定价中的@@ 已售日志。
- Amazon Data Firehose 定价
- Amazon S3 定价

用户活动日志导出包含安全评估，并且是用户池高级安全特征的一项功能。只有在高级安全特征处于活动状态时，Amazon Cognito 才会生成这些日志。这些特征会增加用户池中每月活跃用户（MAU）的费用。有关更多信息，请参阅 Amazon Cognito 定价。

用户活动日志是**INFO**级别的

导出的用户活动日志仅处于INFO错误级别，为身份验证活动的统计和安全分析提供了信息。WARNING和ERROR错误级别的消息（例如限制错误）不包含在导出的日志中。

尽力传输

从 Amazon Cognito 传输日志将尽力而为。您的用户池提供的日志量以及日 CloudWatch 志、Amazon S3 和 Firehose 的服务配额可能会影响日志的传输。

现有的外部日志不受影响

这些日志记录选项不会取代或更改用户池的以下日志功能。

1. CloudTrail 常规用户活动日志，例如注册和登录。
2. 使用 CloudWatch 指标大规模分析用户活动。

另外，您还可以在 Logs 中查找日在 CloudWatch 控制台中查看用户池导入结果志，也可以在 Lo CloudWatch gs 使用 Lambda 触发器自定义用户池工作流 中查找日志。Amazon Cognito 和 Lambda 将这些日志存储在与您为用户活动日志指定的日志组不同的日志组中。

仅适用于用户池

身份池没有日志导出功能。

需要用户权限和服务相关角色

设置日志导出的 Amazon 委托人必须具有修改目标资源的权限，如以下主题所述。Amazon Cognito 代表您创建一个服务相关角色并代入角色，以便向目标资源传输日志。

有关从 Amazon Cognito 发送日志的授权模式的更多信息，请参阅《亚马逊日志用户指南 Amazon Web Services 服务》中的启用 CloudWatch 日志记录。

每种日志类型的日志级别是特定的

消息传递日志为 `userNotification` 类型和 `ERROR` 错误级别。高级安全用户活动日志为 `userAuthEvents` 类型和 `INFO` 错误级别。您可以合并两个成员，一个用于 CloudWatch 日志`LogConfigurations`，一个用于`userNotification`存储 Firehose、Amazon S3 或 CloudWatch 日志。`userAuthEvents`

不能将用户活动日志发送到多个目的地。除了 “日志” 之外 CloudWatch ，您无法将用户通知日志发送到任何目的地。

不同的配置选项

您只能使用 Amazon Cognito 用户池 API 或 Amazon SDK 配置用户通知日志。您可以使用 API 或在 Amazon Cognito 控制台中配置高级安全用户活动日志。要同时设置两者，请使用 API，如中的示例请求所示SetLogDeliveryConfiguration。

基于大型资源的策略需要其他配置

要将日志发送到资源策略大小超过 5120 个字符的日志组，请使用以 `/aws/vendedlogs` 开头的路径配置日志组。有关更多信息，请参阅启用某些 Amazon 服务的日志记录。

在 Amazon S3 中自动创建文件夹

当您配置威胁防护日志导出到 Amazon S3 存储桶时，Amazon Cognito 可能会在您的存储桶中创建一个`AWSLogs`文件夹。并非在所有情况下都创建了该文件夹，并且无需创建即可成功进行配置。

导出电子邮件和短信消息传输错误

对于电子邮件和短信消息传输错误，您可以从用户池中传输 Error 级别的用户通知日志。激活此功能后，您可以选择希望 Amazon Cognito 将日志发送到哪个日志组。当您想了解您的用户池通过 Amazon SNS 和 Amazon SES 传递的电子邮件和短信消息的状态时，用户通知日志记录非常有用。与用户活动导出不同，此日志导出选项不需要 Plus 功能计划。

您可以在 API 请求中使用 Amazon Cognito 用户池 API 配置详细的SetLogDeliveryConfiguration通知日志。您可以在 GetLogDeliveryConfigurationAPI 请求中查看用户池的日志配置。以下是一个示例请求正文。

```
{
   "LogConfigurations": [
```

```
      {
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:example-user-
pool-exported"
         },
         "EventSource": "userNotification",
         "LogLevel": "ERROR"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

您必须使用具有以下权限的 Amazon 证书来授权这些请求。

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "CognitoLog",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
```

```
            "Sid": "CognitoLoggingCWL",
            "Action": [
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

以下是用户池中的一个示例事件。此日志架构可能会发生变化。某些字段可能记录为空值。

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_NOTIFICATION",
    "logLevel": "ERROR",
    "message": {
        "details": "String"
    },
    "logSourceId": {
        "userPoolId": "String"
    }
}
```

导出威胁防护用户活动日志

具有 Plus 功能计划和威胁防护的用户池记录用户活动事件：对用户池中的用户登录、注销和其他身份验证操作的详细信息和安全评估。您可能想要在自己的日志管理系统中查看用户活动日志，或者创建归档。您可以将这些数据导出到亚马逊 CloudWatch 日志组、亚马逊数据 Firehose 流或亚马逊简单存储服务 (Amazon S3) 存储桶。在这里，您可以将这些数据摄取到其他系统中，这些系统以适合您的操作流程的方式分析、标准化或以其他方式处理数据。要导出此类数据，您的用户池必须在 Plus 功能计划中，并且您的用户池中必须启用高级安全功能。

利用这些用户活动日志中的信息，您可以查看用户登录和账户管理活动的概况。默认情况下，Amazon Cognito 会将这些事件捕获到与用户池关联的存储中。以下示例是已登录且评估结果表明没有风险因素的用户的示例事件。您可以使用 `AdminListUserAuthEvents` API 操作来检索此信息。下面是一个示例输出：

```
{
    "AuthEvents": [
        {
            "EventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
            "EventType": "SignIn",
            "CreationDate": "2024-06-27T10:49:59.139000-07:00",
            "EventResponse": "Pass",
            "EventRisk": {
                "RiskDecision": "NoRisk",
                "CompromisedCredentialsDetected": false
            },
            "ChallengeResponses": [
                {
                    "ChallengeName": "Password",
                    "ChallengeResponse": "Success"
                }
            ],
            "EventContextData": {
                "IpAddress": "192.0.2.1",
                "DeviceName": "Chrome 126, Windows 10",
                "Timezone": "-07:00",
                "City": "null",
                "Country": "United States"
            }
        }
    ],
    "NextToken": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222#2024-06-27T17:49:59.139Z"
}
```


您可以在 Amazon Cognito 控制台中或SetLogDeliveryConfiguration通过 API 操作为用户活动激活日志导出。

Amazon Web Services Management Console

1. 如果您还没有要使用的存储桶，请创建 S3 存储桶、Firehose 流或CloudWatch日志组。
2. 登录 Amazon Cognito 控制台。
3. 选择用户池。
4. 从列表中选择一个现有用户池，或创建一个用户池。
5. 选择高级安全选项卡。找到导出用户活动日志，然后选择编辑。
6. 在日志记录状态下，选中激活用户活动日志导出旁边的复选框。

7. 在日志目标下，选择要处理日志 Amazon Web Services 服务 的：CloudWatch 日志组、Amazon Data Firehose 流或 S3 存储桶。
8. 您的选择将使用相应的资源类型填入资源选择器。从列表中选择日志组、流或存储桶。您也可以选择 “创建” 按钮，导航到 Amazon Web Services Management Console 所选服务的，然后创建新资源。
9. 选择保存更改。

API

为您的用户活动日志选择一种目标类型。

以下是将 Firehose 流设置为日志目标的示例 `SetLogDeliveryConfiguration` 请求正文。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "FirehoseConfiguration": {
            "StreamArn": "arn:aws:firehose:us-west-2:123456789012:deliverystream/
example-user-pool-activity-exported"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

以下是将 Amazon S3 存储桶设置为日志目标的示例 `SetLogDeliveryConfiguration` 请求正文。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "S3Configuration": {
            "BucketArn": "arn:aws:s3:::amzn-s3-demo-logging-bucket"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
```

```
}
```

以下是将 CloudWatch 日志组设置为日志目标的`SetLogDeliveryConfiguration`请求正文示例。

```
{
   "LogConfigurations": [
      {
         "EventSource": "userAuthEvents",
         "CloudWatchLogsConfiguration": {
            "LogGroupArn": "arn:aws:logs:us-west-2:123456789012:log-group:DOC-
EXAMPLE-LOG-GROUP"
         },
         "LogLevel": "INFO"
      }
   ],
   "UserPoolId": "us-west-2_EXAMPLE"
}
```

配置日志传送的用户必须是用户池管理员并具有以下额外权限：

Amazon S3

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageLogsS3",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
```

```
                "s3:PutBucketPolicy",
                "s3:GetBucketPolicy"
            ],
            "Resource": "*"
        }
    ]
}
```

CloudWatch Logs

```
{
    "Version": "2012-10-17",
    "Statement": [
            {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageLogsCWL",
            "Action": [
                "logs:CreateLogDelivery",
                "logs:GetLogDelivery",
                "logs:UpdateLogDelivery",
                "logs:DeleteLogDelivery",
                "logs:ListLogDeliveries",
                "logs:PutResourcePolicy",
                "logs:DescribeResourcePolicies",
                "logs:DescribeLogGroups"
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        }
    ]
}
```

Amazon Data Firehose

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "ManageUserPoolLogs",
            "Action": [
                "cognito-idp:SetLogDeliveryConfiguration",
                "cognito-idp:GetLogDeliveryConfiguration",
            ],
            "Resource": [
                "*"
            ],
            "Effect": "Allow"
        },
        {
            "Sid": "ManageUserPoolLogsFirehose",
            "Effect": "Allow",
            "Action": [
                "logs:CreateLogDelivery",
                "iam:CreateServiceLinkedRole",
                "firehose:TagDeliveryStream"
            ],
            "Resource": "*"
        }
    ]
}
```

以下是用户池中的一个示例事件。此日志架构可能会发生变化。某些字段可能记录为空值。

```
{
    "eventTimestamp": "1687297330677",
    "eventSource": "USER_ACTIVITY",
    "logLevel": "INFO",
    "message": {
        "version": "1",
        "eventId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE11111",
        "eventType": "SignUp",
        "userSub": "a1b2c3d4-5678-90ab-cdef-EXAMPLE22222",
        "userName": "test-user",
        "userPoolId": "us-west-2_EXAMPLE",
```

```
        "clientId": "1example23456789",
        "creationDate": "Wed Jul 17 17:25:55 UTC 2024",
        "eventResponse": "InProgress",
        "riskLevel": "",
        "riskDecision": "PASS",
        "challenges": [],
        "deviceName": "Other, Other",
        "ipAddress": "192.0.2.1",
        "requestId": "a1b2c3d4-5678-90ab-cdef-EXAMPLE33333",
        "idpName": "",
        "compromisedCredentialDetected": "false",
        "city": "Seattle",
        "country": "United States",
        "eventFeedbackValue": "",
        "eventFeedbackDate": "",
        "eventFeedbackProvider": "",
        "hasContextData": "true"
    },
    "logSourceId": {
        "userPoolId": "us-west-2_EXAMPLE"
    }
}
```

跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况

您可以使用亚马逊 CloudWatch 或服务配额来监控 Amazon Cognito 用户池。您还可以在 Service Quotas 中监控身份池的使用情况。 CloudWatch 收集原始数据并将其处理成可读的、近乎实时的指标。在中 CloudWatch，您可以设置警报，监视某些阈值，并在达到这些阈值时发送通知或采取行动。要为服务配额创建 CloudWatch 警报，请参阅创建 CloudWatch警报。Amazon Cognito 指标每隔五分钟提供一次。有关中保留期限的更多信息 CloudWatch，请访问 Amazon CloudWatch 常见问题页面。

您可以使用 Service Quotas 查看和管理 Amazon Cognito 用户池及身份池配额使用情况。Service Quotas 控制台具有三个功能：查看服务配额、请求提高服务配额以及查看当前利用率。您可以使用第一个功能来查看配额，并查看配额是否可调节。您可以使用第二个功能请求增加 Service Quotas。您可以使用最后一个功能查看配额利用率。此功能仅在您的账户已激活一段时间后才可用。有关在 Service Quotas 控制台中查看配额的更多信息，请参阅查看 Service Quotas。

> **Note**
>
> Amazon Cognito 指标每 5 分钟提供一次。有关中保留期限的更多信息 CloudWatch，请访问 Amazon CloudWatch 常见问题页面。

如果您登录 Amazon Web Services 账户 的是设置为 CloudWatch跨账户可观察性的监控账户，则可以使用该监控账户来可视化服务配额，并为与该监控账户关联的源账户中的指标设置警报。有关更多信息，请参阅 CloudWatch 跨账户可观测性。

主题

- 中的用户池指标 CloudWatch
- Service Quotas 中的指标

中的用户池指标 CloudWatch

用户池将用户活动统计数据报告 CloudWatch 为指标。您可以从中 CloudWatch分析用户池中的身份验证活动量和配额使用情况。利用这些指标中的信息，您可为值得注意的事件设置警报，并根据需要调整用户池配置。用户活动日志记录包含用户池中用户活动的详细记录，而 CloudWatch 指标则包含汇总的统计数据和绩效指标。

下表列出了对 Amazon Cognito 用户池可用的指标。Amazon Cognito 将指标发布到命名空间和`AWS/Cognito`. `AWS/Usage` 有关更多信息，请参阅 A mazon CloudWatch 用户指南中的命名空间。

有关跟踪配额和使用情况的更多信息，请参阅跟踪配额使用量和跟踪每月活跃用户 (MAUs)。

> **Note**
>
> 控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台中所有指标选项卡的搜索框中输入其指标名称或维度名称时，它们也不会显示。此外，它们不会在 list-metrics 命令的结果中返回。检索这些指标的最佳方法是在 Amazon CLI 中使用`get-metric-data`或`get-metric-statistics`命令。

指标	描述	命名空间
`SignUpSuccesses`	提供向 Amazon Cognito 用户池发出的成功用户注册请求的总数。一个成功的用户注册请求会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。	`AWS/Cognito`

指标	描述	命名空间
	要查找成功的用户注册请求的百分比，请对此指标使用 Average 统计数据。要计算用户注册请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的用户注册请求的总数，请对此指标使用 Sum 统计数据。要计算失败的用户注册请求总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计Sample Count数据。 为每个用户池客户端的用户池发布此指标。如果用户注册由管理员执行，则以 Admin 身份将指标与用户池客户端一起发布。 请注意，不会针对用户导入和用户迁移案例发出此指标。 指标维度：UserPool、UserPoolClient 单位：计数	

指标	描述	命名空间
SignUpThrottles	提供向 Amazon Cognito 用户池发出的受限的用户注册请求的总数。当用户注册请求受到限制时，将发布计数 1。 要计算受限制的用户注册请求的总数，请对此指标使用 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果受限制的请求由管理员发出，则以 Admin 身份将指标与用户池客户端一起发布。 指标维度：UserPool、UserPoolClient 单位：计数	AWS/Cognito

指标	描述	命名空间
SignInSuccesses	提供向 Amazon Cognito 用户池发出的成功的用户身份验证请求的总数。在向用户颁发身份验证令牌时，用户身份验证将被视为成功。一个成功的身份验证会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。 要查找成功的用户身份验证请求的百分比，请对此指标使用 Average 统计数据。要计算用户身份验证请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的用户身份验证请求的总数，请对此指标使用 Sum 统计数据。要计算失败的用户身份验证请求总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计Sample Count数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 请注意，Amazon Cognito 令牌刷新请求并未包含在此指标	AWS/Cognito

指标	描述	命名空间
	中。有一个用于提供 Refresh 令牌统计数据的单独指标。 指标维度：UserPool、UserPoolClient 单位：计数	
SignInThrottles	提供向 Amazon Cognito 用户池发出的受限制的用户身份验证请求的总数。当身份验证请求受到限制时，将发布计数 1。 要计算受限制的用户身份验证请求的总数，请对此指标使用 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 Amazon Cognito 令牌刷新请求并未包含在此指标中。有一个用于提供 Refresh 令牌统计数据的单独指标。 指标维度：UserPool、UserPoolClient 单位：计数	AWS/Cognito

指标	描述	命名空间
TokenRefreshSuccesses	提供向 Amazon Cognito 用户池发出的成功的 Amazon Cognito 令牌刷新请求的总数。一个成功的 Amazon Cognito 令牌刷新请求会产生值 1，而一个不成功的请求会产生值 0。受限制的请求也会被视为不成功的请求，因此，一个受限制的请求也将产生计数 0。 要查找成功的 Amazon Cognito 令牌刷新请求的百分比，请对此指标使用 Average 统计数据。要计算 Amazon Cognito 令牌刷新请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的 Amazon Cognito 令牌刷新请求的总数，请对此指标使用Sum统计数据。要计算刷新 Amazon Cognito 令牌的失败请求总数，请使用 CloudWatchMath表达式并从Sum统计数据中减去统计数据。Sample Count 按每个用户池客户端发布此指标。如果请求中有无效的用户池客户端，则用户池客户端值包含固定值 Invalid。 指标维度：UserPool、UserPoolClient	AWS/Cognito

指标	描述	命名空间
	单位：计数	
TokenRefreshThrottles	提供向 Amazon Cognito 用户池发出的受限制 Amazon Cognito 令牌刷新请求的总数。当 Amazon Cognito 令牌刷新请求受到限制时，将发布计数 1。 要计算受限制的 Amazon Cognito 令牌刷新请求的总数，请对此指标使用 Sum 统计数据。 为每个客户端的每个用户池发布此指标。如果请求中提供了无效的用户池客户端，则指标中的相应用户池客户端值将包含固定值 Invalid，而不是请求中发送的实际无效值。 指标维度：UserPool、UserPoolClient 单位：计数	AWS/Cognito

指标	描述	命名空间
FederationSuccesses	提供向 Amazon Cognito 用户池发出的成功的联合身份验证请求的总数。当 Amazon Cognito 向用户颁发身份验证令牌时，身份联合验证被视为成功。一个成功的联合身份验证请求会产生值 1，而一个不成功的请求会产生值 0。节流的请求以及生成授权码但没有令牌的请求所生成的值为 0。 要查找成功的联合身份验证请求的百分比，请对此指标使用 Average 统计数据。要计算联合身份验证请求的总数，请对此指标使用 Sample Count 统计数据。要计算成功的联合身份验证请求的总数，请对此指标使用 Sum 统计数据。要计算失败的身份联合请求总数，请使用 CloudWatch Math表达式并从Sum统计数据中减去统计Sample Count数据。 指标维度：UserPool、UserPoolClient 、IdentityProvider 单位：计数	AWS/Cognito

指标	描述	命名空间
FederationThrottles	提供向 Amazon Cognito 用户池发出的受限制的联合身份验证请求的总数。当联合身份验证请求受到限制时，将发布计数 1。 要计算受限制的联合身份验证请求的总数，请对此指标使用 Sum 统计数据。 指标维度：UserPool、UserPoolClient 、IdentityProvider 单位：计数	AWS/Cognito
CallCount	提供客户发出的与类别相关的调用总数。此指标包括所有调用，如受限制的调用、失败的调用和成功的调用。 在账户和地区的所有用户池中，每个 Amazon 账户都必须使用类别配额。 您可以使用此指标的 Sum 统计数据计算调用总数。 指标维度：服务、类型、资源、类 单位：计数	AWS/Usage

指标	描述	命名空间
ThrottleCount	提供与类别相关的受限调用总数。 此指标已在账户级别发布。 您可以使用此指标的 Sum 统计数据计算某个类别中的调用总数。 指标维度：服务、类型、资源、类 单位：计数	AWS/Usage

查看威胁防护指标

您的用户池发布的指标包含有关以下方面的统计信息：您的威胁防护设置对用户身份验证活动的影响。您可能想知道有多少用户尝试使用已泄露的凭证登录。您还可以了解有多少百分比的登录活动被评估为存在一定风险。Amazon Cognito 将威胁防护功能的指标发布到您在亚马逊上的账户。CloudWatchAmazon Cognito 按风险级别和请求级别将威胁防护指标组合在一起。

要为风险分析添加背景信息，您可以在用户池或导出的数据来源中查看有关单个用户登录尝试的信息。

在 CloudWatch 控制台中查看指标

1. 打开 CloudWatch 控制台，网址为https://console.aws.amazon.com/cloudwatch/。
2. 在导航窗格中，选择指标。
3. 选择 Amazon Cognito。
4. 选择一组聚合指标，如按风险分类。
5. 所有指标选项卡显示该选择的所有指标。您可执行以下操作：
   - 要对表进行排序，请使用列标题。
   - 要为指标绘制图表，请选中该指标旁的复选框。要选择所有指标，请选中表的标题行中的复选框。
   - 要按资源进行筛选，请选择资源 ID，然后选择添加到搜索。
   - 要按指标进行筛选，请选择指标名称，然后选择添加到搜索。

指标	描述	指标维度	命名空间
CompromisedCredentialRisk	Amazon Cognito 在其中检测到泄露凭证的请求。	Operation：操作类型。仅有的维度是 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。 RiskLevel：高（默认）、中或低。	`AWS/Cognito`
AccountTakeoverRisk	Amazon Cognito 在其中检测到账户接管风险的请求。	Operation：操作类型。仅有的维度是 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。 RiskLevel: 高、中或低。	`AWS/Cognito`
OverrideBlock	因开发人员提供的配置而被 Amazon Cognito 阻止的请求。	Operation：操作类型。仅有的维度是 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。 RiskLevel: 高、中或低。	`AWS/Cognito`

指标	描述	指标维度	命名空间
Risk	Amazon Cognito 标记为有风险的请求。	Operation：操作类型，例如 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。	`AWS/Cognito`
NoRisk	Amazon Cognito 在其中没有识别出任何风险的请求。	Operation：操作类型，例如 `PasswordChange` 、`SignIn` 或 `SignUp`。 UserPoolId：用户池的标识符。	`AWS/Cognito`

Amazon Cognito 为您提供了两组预定义的指标，供您随时进行分析。 CloudWatch按风险分类标识 Amazon Cognito 认定为有风险的请求的风险级别。按请求分类体现了按请求级别聚合的指标。

汇总指标组	描述
按风险分类	被 Amazon Cognito 标识为有风险的请求。
按请求分类	按请求汇总的指标。

Amazon Cognito 用户池的维度

以下维度用于优化由 Amazon Cognito 发布的用量指标。维度仅适用于 `CallCount` 和 `ThrottleCount` 指标。

维度	描述
服务	包含资源的 Amazon 服务的名称。对于 Amazon Cognito 用量指标，此维度的值为 `Cognito user pool`。

维度	描述
类型	正在报告的实体的类型。Amazon Cognito 用量指标的唯一有效值为 API。
资源	正在运行的资源的类型。唯一的有效值是类别名。
类	所跟踪的资源的类。Amazon Cognito 不使用类维度。

使用 CloudWatch 控制台跟踪指标

您可以使用跟踪和收集 Amazon Cognito 用户池指标。 CloudWatch CloudWatch 控制面板将显示有关您使用的每项 Amazon 服务的指标。您可以使用 CloudWatch 创建指标警报。可以将警报设置为向您发送通知或更改您正在监控的特定资源。要在中查看服务配额指标 CloudWatch，请完成以下步骤。

1. 打开 CloudWatch 管理控制台。
2. 在导航窗格中，选择指标。
3. 在所有指标中，选择一个指标和维度。
4. 选中指标旁边的复选框。指标将出现在图表中。

> **Note**
>
> 控制台中不会显示在过去两周内没有任何新数据点的指标。当您在控制台的“全部指标”选项卡的搜索框中输入指标名称或维度名称时，它们也不会显示，并且 list-metrics 命令的结果中不会返回它们。检索这些指标的最佳方法是在 Amazon CLI 中使用 `get-metric-data` 或者 `get-metric-statistics` 命令。

为配额创建 CloudWatch 警报

Amazon Cognito 提供的 CloudWatch 使用量指标与和的 Amazon 服务配额`CallCount`相对应。`ThrottleCount` APIs有关在中跟踪使用情况的更多信息 CloudWatch，请参阅跟踪配额使用量。

在 Service Quotas 控制台中，您可以创建告警，以便在您的使用量接近服务配额时提示您。要了解如何使用 Service Quotas 控制台设置 CloudWatch 警报，请参阅服务配额和 CloudWatch 警报。

Service Quotas 中的指标

借助 Service Quotas，您可以从一个中心位置查看和管理 Amazon Cognito 用户池及身份池配额。您可以使用 Service Quotas 控制台查看具体配额的详细信息、监控配额利用率以及请求增加配额。对于某些配额类型，您可以创建 CloudWatch警报来跟踪您的配额使用情况。要详细了解您可以跟踪哪些 Amazon Cognito 指标，请参阅跟踪配额使用量。

要查看 Amazon Cognito 用户池和身份池的服务限额使用情况，请完成以下步骤。

1. 打开服务限额控制台。
2. 在导航窗格中，选择 Amazon 服务。
3. 从 Amazon 服务列表中，搜索并选择 Amazon Cognito 用户池或 Amazon Cognito 联合身份。此时将显示服务配额页面。
4. 选择支持 CloudWatch 监控的配额。例如，在 Amazon Cognito 用户池中选择 `Rate of UserAuthentication requests`。
5. 向下滚动到监控。此部分仅针对支持 CloudWatch 监控的配额显示。
6. 在监控中，您可以在图表中查看当前服务配额利用率。
7. 在监控)中，选择 1 小时、3 小时、12 小时、1 天、3 天或 1 周。
8. 选择图表中的任意区域，以查看服务配额利用率百分比。在这里，您可以将图表添加到控制面板或使用操作菜单选择在指标中查看，这将带您进入 CloudWatch 控制台中的相关指标。

亚马逊 Cognito 正在登录 Amazon CloudTrail

Amazon Cognito 与 Amazon CloudTrail一项服务集成，可记录用户、角色或 Amazon 服务在 Amazon Cognito 中执行的操作。 CloudTrail 捕获一部分 Amazon Cognito 的 API 调用作为事件，包括来自亚马逊 Cognito 控制台的调用和对亚马逊 Cognito API 操作的代码调用。如果您创建了跟踪，则可以选择将 CloudTrail 事件传送到 Amazon S3 存储桶，包括 Amazon Cognito 的事件。如果您未配置跟踪，您仍然可以在 CloudTrail 控制台的 “事件历史记录” 中查看最新的事件。通过收集的信息 CloudTrail，您可以确定向 Amazon Cognito 发出的请求、发出请求的 IP 地址、谁提出了请求、何时提出请求以及其他详细信息。

要了解更多信息 CloudTrail，包括如何配置和激活它，请参阅Amazon CloudTrail 用户指南。

您还可以为特定 CloudTrail 事件创建 Amazon CloudWatch 警报。例如，您可以设置 CloudWatch ，以在身份池配置发生更改时触发警报。有关更多信息，请参阅为 CloudTrail 事件创建 CloudWatch 警报：示例。

主题

- 亚马逊 Cognito 发送到的信息 CloudTrail
- 使用亚马逊日志见解分析 Amazon Cognito CloudTrail 事件 CloudWatch
- 示例 Amazon Cognito 事件

亚马逊 Cognito 发送到的信息 CloudTrail

CloudTrail 在您创建时已开启 Amazon Web Services 账户。当 Amazon Cognito 中出现支持的事件活动时，该活动会与其他 Amazon 服务 CloudTrail 事件一起记录在事件历史记录中。您可以在自己的 Amazon 账户中查看、搜索和下载最近发生的事件。有关更多信息，请参阅使用事件历史查看 CloudTrail 事件。

要持续记录您的 Amazon 账户中的事件，包括 Amazon Cognito 的事件，请创建跟踪。 CloudTrail 跟踪将日志文件传输到 Amazon S3 存储桶。预设情况下，在控制台中创建跟踪时，此跟踪应用于所有 区域。跟踪记录 Amazon 分区中所有区域的事件，并将日志文件传送到您指定的 Amazon S3 存储桶。此外，您可以配置其他 Amazon 服务，以进一步分析和处理 CloudTrail 日志中收集的事件数据。有关更多信息，请参阅：

- 创建跟踪概览
- CloudTrail 支持的服务和集成
- 为配置亚马逊 SNS 通知 CloudTrail
- 接收来自多个地区的 CloudTrail 日志文件和接收来自多个账户的 CloudTrail日志文件

每个事件或日志条目都包含有关生成请求的人员信息。身份信息可帮助您确定以下内容：

- 请求是使用根用户凭证还是 IAM 用户凭证发出的。
- 请求是使用角色还是联合用户的临时安全凭证发出的。
- 请求是否由其他 Amazon 服务发出。

有关更多信息，请参阅 CloudTrail userIdentity 元素。

中的机密数据 Amazon CloudTrail

由于用户池和身份池会处理用户数据，因此 Amazon Cognito 会使用该值掩盖 CloudTrail 事件中的一些私有字段。`HIDDEN_FOR_SECURITY_REASONS`有关 Amazon Cognito 未填充到事件的字段的示

例，请参阅示例 Amazon Cognito 事件。Amazon Cognito 只会掩盖一些通常包含用户信息的字段，例如密码和令牌。Amazon Cognito 不会自动检测或屏蔽您在 API 请求中填充到非私有字段的个人身份信息。

用户池事件

Amazon Cognito 支持将用户池操作页面上列出的所有操作作为事件 CloudTrail 记录在日志文件中。Amazon Cognito 将用户池事件记录 CloudTrail 为管理事件。

Amazon Cognito 用户池 CloudTrail 条目中的`eventType`字段告诉你你的应用程序是向 A mazon Cognito 用户池 API 发出请求，还是向为 OpenID Connect、SAML 2.0 或托管登录页面提供资源的终端节点发出了请求。API 请求的 `AwsApiCall` 为 `eventType`，端点请求的 `AwsServiceEvent` 为 `eventType`。

Amazon Cognito 将以下请求作为事件记录到您的托管登录服务。 CloudTrail

Hosted UI (classic) events

在中托管用户界面（经典）活动 CloudTrail

操作	描述
`Login_GET`，`CognitoAuthentication`	用户查看或向您的 登录端点 提交凭证。
`OAuth2_Authorize_GET`，`Beta_Authorize_GET`	用户查看您的 对端点授权。
`OAuth2Response_GET`，`OAuth2Response_POST`	用户向您的 `/oauth2/idpresponse` 端点提交 IdP 令牌。
`SAML2Response_POST`，`Beta_SAML2Response_POST`	用户向您的 `/saml2/idpresponse` 端点提交 IdP SAML 断言。
`Login_OIDC_SAML_POST`	用户在您的 登录端点 中输入用户名并与 IdP 标识符匹配。
`Token_POST`，`Beta_Token_POST`	用户向您的 令牌端点 提交授权码。
`Signup_GET`，`Signup_POST`	用户向您的 `/signup` 端点提交注册信息。
`Confirm_GET`，`Confirm_POST`	用户在托管 UI 中提交确认代码。

操作	描述
ResendCode_POST	用户在托管 UI 中提交重新发送确认代码的请求。
ForgotPassword_GET ，ForgotPassword_POST	用户向您的 /forgotPassword 端点提交重置密码的请求。
ConfirmForgotPassword_GET ，ConfirmForgotPassword_POST	用户向您的 /confirmForgotPassword 端点提交代码以确认其 ForgotPassword 请求。
ResetPassword_GET ，ResetPassword_POST	用户在托管 UI 中提交新密码。
Mfa_GET, Mfa_POST	用户在托管 UI 中提交多重身份验证（MFA）代码。
MfaOption_GET ，MfaOption_POST	用户在托管 UI 中选择其首选 MFA 方法。
MfaRegister_GET ，MfaRegister_POST	用户在注册 MFA 时，在托管 UI 中提交多重身份验证（MFA）代码。
Logout	用户在您的 /logout 端点注销。
SAML2Logout_POST	用户在您的 /saml2/logout 端点注销。
Error_GET	用户在托管 UI 中查看错误页面。
UserInfo_GET ，UserInfo_POST	用户或 IdP 与您的 userInfo 端点 交换信息。
Confirm_With_Link_GET	用户根据 Amazon Cognito 在电子邮件中发送的链接提交确认。
Event_Feedback_GET	用户向 Amazon Cognito 提交有关高级安全功能事件的反馈。

Managed login events

中的托管登录事件 CloudTrail

操作	描述
`login_POST`	用户向您的提交凭证登录端点。
`login_continue_POST`	已经登录过一次的用户选择再次登录。
`forgotPassword_POST`	用户重置密码。
`selectChallenge_POST`	用户在提交用户名或凭据后对身份验证质询作出回应。
`confirmUser_GET`	用户在确认或验证电子邮件中打开链接。
`mfa_back_POST`	用户在出现 MFA 提示后选择 “返回” 按钮。
`mfa_options_POST`	用户选择 MFA 选项。
`mfa_phone_register_POST`	用户提交电话号码以注册为 MFA 因子。此操作会导致 Amazon Cognito 向他们的电话号码发送 MFA 代码。
`mfa_phone_verify_POST`	用户提交发送到其电话号码的 MFA 代码。
`mfa_phone_resendCode_POST`	用户提交了向其电话号码重新发送 MFA 代码的请求。
`mfa_totp_POST`	用户提交 TOTP MFA 代码。
`signup_POST`	用户向您的`/signup`托管登录页面提交信息。
`signup_confirm_POST`	用户通过电子邮件或短信提交确认码。
`verifyCode_POST`	用户提交一次性密码 (OTP) 以进行无密码身份验证。
`passkeys_add_POST`	用户提交了注册新密钥凭证的请求。
`passkeys_add_GET`	用户导航到可以注册密钥的页面。

操作	描述
`login_passkey_POST`	用户使用密钥登录。

> **Note**
>
> Amazon Cognito 会记录特定`UserName`于用户的请求，`UserSub`但不记录在 CloudTrail 日志中。通过调用 `ListUsers` API，并使用主题筛选条件，您可以找到给定 `UserSub` 的用户。

身份池事件

数据事件

Amazon Cognito 将以下 Amazon Cognito 身份事件记录 CloudTrail 为数据事件。数据事件是大容量数据平面 API 操作， CloudTrail 默认情况下不记录。记录数据事件将收取额外费用。

- GetCredentialsForIdentity
- GetId
- GetOpenIdToken
- GetOpenIdTokenForDeveloperIdentity
- UnlinkIdentity

要为这些 API 操作生成 CloudTrail 日志，您必须激活跟踪中的数据事件，并为 Cognito 身份池选择事件选择器。有关更多信息，请参阅 Amazon CloudTrail 用户指南中的记录数据事件以便跟踪。

您还可以使用以下 CLI 命令将身份池事件选择器添加到您的跟踪记录中。

```
aws cloudtrail put-event-selectors --trail-name <trail name> --advanced-event-selectors
 \
"{\
   \"Name\": \"Cognito Selector\",\
   \"FieldSelectors\": [\
      {\
         \"Field\": \"eventCategory\",\
         \"Equals\": [\
            \"Data\"\
         ]\
```

```
      },\
      {\
         \"Field\": \"resources.type\",\
         \"Equals\": [\
            \"AWS::Cognito::IdentityPool\"\
         ]\
      }\
   ]\
}"
```

管理事件

Amazon Cognito 将剩余的 Amazon Cognito 身份池 API 操作记录为管理事件。 CloudTrail 默认情况下会记录管理事件 API 操作。

有关 Amazon Cognito 登录 CloudTrail的亚马逊 Cognito 身份池 API 操作的列表，请参阅亚马逊 Cognito 身份池 API 参考。

Amazon Cognito Sync

Amazon Cognito 将所有 Amazon Cognito 同步 API 操作记录为管理事件。有关 Amazon Cognito 登录 CloudTrail的亚马逊 Cognito Sync API 操作的列表，请参阅亚马逊 Cognito Sync API 参考。

使用亚马逊日志见解分析 Amazon Cognito CloudTrail 事件 CloudWatch

您可以使用 Amazon Log CloudWatch s Insights 搜索和分析您的 Amazon Cognito CloudTrail 事件。当您将跟踪配置为向 CloudWatch Logs 发送事件时，仅 CloudTrail 发送与您的跟踪设置相匹配的事件。

要查询或研究您的 Amazon Cognito CloudTrail 事件，请在 CloudTrail 控制台中确保在跟踪设置中选择管理事件选项，以便您可以监控对资源执行的管理操作。 Amazon 当您想要识别账户中的错误、异常活动或异常用户行为时，可以在跟踪记录设置中选择 Insights 事件选项。

Amazon Cognito 查询的示例

您可以在 Amazon CloudWatch 控制台中使用以下查询。

常规查询

查找 25 个最近添加的日志事件。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
```

```
| filter eventSource = "cognito-idp.amazonaws.com"
```

获取 25 个最近添加的录入事件（包含异常）的列表。

```
fields @timestamp, @message | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and @message like /Exception/
```

异常和错误查询

查找最近添加的 25 个含错误代码 `NotAuthorizedException` 的录入事件以及 Amazon Cognito 用户池 `sub`。

```
fields @timestamp, additionalEventData.sub as user | sort @timestamp desc | limit 25
| filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
```

查找具有 `sourceIPAddress` 和相应 `eventName` 的记录数。

```
filter eventSource = "cognito-idp.amazonaws.com"
| stats count(*) by sourceIPAddress, eventName
```

查找触发 `NotAuthorizedException` 错误的前 25 个 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and errorCode=
 "NotAuthorizedException"
| stats count(*) as count by sourceIPAddress, eventName
| sort count desc | limit 25
```

找到调用 `ForgotPassword` API 的前 25 个 IP 地址。

```
filter eventSource = "cognito-idp.amazonaws.com" and eventName = 'ForgotPassword'
| stats count(*) as count by sourceIPAddress
| sort count desc | limit 25
```

示例 Amazon Cognito 事件

Amazon Cognito 将 Amazon CloudTrail 有关用户身份验证活动和管理管理活动的信息记录到。这适用于用户池和身份池。例如，您可以查看同一跟踪中的 `GetId` 和 `UpdateIdentityPool` 事件，或者 `UpdateAuthEventFeedback` 和 `SetRiskConfiguration` 事件。您还会将看到与用户池 API 中

的操作不对应的托管 UI 活动的用户池日志。这一部分提供了一些您在实际使用中可能会看到的日志示例。要了解任何操作 CloudTrail 的事件架构，请为该操作生成请求并查看该操作在您的跟踪中创建的事件。

跟踪可以将事件作为日志文件传输到您指定的 Amazon S3 存储桶。 CloudTrail 日志文件包含一个或多个日志条目。事件代表来自任何来源的单个请求，包括有关请求的操作、操作的日期和时间、请求参数等的信息。 CloudTrail 日志文件不是公共 API 调用的有序堆栈跟踪，因此它们不会按任何特定顺序出现。

主题

- 托管用户界面注册的示例 CloudTrail 事件
- SAML 请求的示例 CloudTrail 事件
- 向令牌端点发出请求 CloudTrail 的事件示例
- 的示例 CloudTrail 事件 CreateIdentityPool
- 的示例 CloudTrail 事件 GetCredentialsForIdentity
- 的示例 CloudTrail 事件 GetId
- 的示例 CloudTrail 事件 GetOpenIdToken
- 的示例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity
- 的示例 CloudTrail 事件 UnlinkIdentity

托管用户界面注册的示例 CloudTrail 事件

以下示例 CloudTrail 事件演示了用户通过托管用户界面注册时 Amazon Cognito 记录的信息。

当新用户导航到应用程序的登录页面时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-04-06T05:38:12Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Login_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
```

```
    "errorCode": "",
    "errorMessage": "",
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200.0
        },
        "requestParameters":
        {
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "response_type":
            [
                "token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        }
    },
    "eventID": "382ae09a-151d-4116-8f2b-6ac0a804a38c",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当新用户从应用程序的登录页面选择注册时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
```

```
    },
    "eventTime": "2022-05-05T23:21:43Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "7a63e7c2-b057-4f3d-a171-9d9113264fff",
    "eventID": "5e7b27a0-6870-4226-adb4-f86cd51ac5d8",
    "readOnly": true,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
```

```
}
```

当新用户选择用户名、输入电子邮件地址并从应用程序的登录页面选择密码时，Amazon Cognito 会记录以下事件。Amazon Cognito 不会将有关用户身份的识别信息记录到。 CloudTrail

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Signup_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "password":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "requiredAttributes[email]":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
```

```
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "9ad58dd8-3517-4aa8-96a5-d17a01df9eb4",
    "eventID": "c75eb7a5-eb8c-43d1-8331-f4412e756e69",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当新用户在注册后访问托管 UI 中的用户确认页面时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:22:06Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_GET",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
```

```
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "response_type":
            [
                "code"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "58a5b170-3127-45bb-88cc-3e652d779e0b",
    "eventID": "7f87291a-6d50-409a-822f-e3a5ec7e60da",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当用户在托管 UI 的用户确认页面中输入 Amazon Cognito 通过电子邮件发送给他们的代码时，Amazon Cognito 会记录以下事件。

```
{
```

```
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-05T23:23:32Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Confirm_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "confirm":
            [
                ""
            ],
            "deliveryMedium":
            [
                "EMAIL"
            ],
            "sub":
            [
                "704b1e47-34fe-40e9-8c41-504997494531"
            ],
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "destination":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "response_type":
            [
                "code"
```

```
            ],
            "_csrf":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "cognitoAsfData":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "redirect_uri":
            [
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ],
            "username":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "9764300a-ed35-4f87-8a0f-b18b3fe2b11e",
    "eventID": "e24ac6e5-2f70-4c6e-ad4e-2f08a547bb36",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

SAML 请求的示例 CloudTrail 事件

当使用您的 SAML IdP 进行身份验证的用户将 SAML 断言提交给您的 `/saml2/idpresponse` 端点时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-06T00:50:57Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "SAML2Response_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 302
        },
        "requestParameters":
        {
            "RelayState":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "SAMLResponse":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "4f6f15d1-c370-4a57-87f0-aac4817803f7",
    "eventID": "9824b50f-d9d1-4fb8-a2c1-6aa78ca5902a",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "625647942648",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
```

```
    },
    "eventCategory": "Management"
}
```

向令牌端点发出请求 CloudTrail 的事件示例

以下是来自对 令牌端点 的请求的示例事件。

当已通过身份验证并收到授权代码的用户将代码提交到您的 /oauth2/token 端点时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:12:30Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "code":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "authorization_code"
            ],
            "redirect_uri":
            [
```

```
                "https://www.amazon.com"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "f257f752-cc14-4c52-ad5b-152a46915238",
    "eventID": "0bd1586d-cd3e-4d7a-abaf-fd8bfc3912fd",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当您的后端系统向您的 `/oauth2/token` 端点提交访问令牌的 `client_credentials` 请求时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T21:07:05Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
```

```
        {
            "status": 200
        },
        "requestParameters":
        {
            "grant_type":
            [
                "client_credentials"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "4f871256-6825-488a-871b-c2d9f55caff2",
    "eventID": "473e5cbc-a5b3-4578-9ad6-3dfdcb8a6d34",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

当您的应用程序与您的 `/oauth2/token` 端点交换刷新令牌以获取新 ID 和访问令牌时，Amazon Cognito 会记录以下事件。

```
{
    "eventVersion": "1.08",
    "userIdentity":
    {
        "accountId": "123456789012"
    },
    "eventTime": "2022-05-12T22:16:40Z",
    "eventSource": "cognito-idp.amazonaws.com",
    "eventName": "Token_POST",
    "awsRegion": "us-west-2",
```

```
    "sourceIPAddress": "192.0.2.1",
    "userAgent": "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_15_7)...",
    "requestParameters": null,
    "responseElements": null,
    "additionalEventData":
    {
        "responseParameters":
        {
            "status": 200
        },
        "requestParameters":
        {
            "refresh_token":
            [
                "HIDDEN_DUE_TO_SECURITY_REASONS"
            ],
            "grant_type":
            [
                "refresh_token"
            ],
            "client_id":
            [
                "1example23456789"
            ]
        },
        "userPoolDomain": "mydomain.auth.us-west-2.amazoncognito.com",
        "userPoolId": "us-west-2_EXAMPLE"
    },
    "requestID": "2829f0c6-a3a9-4584-b046-11756dfe8a81",
    "eventID": "12bd3464-59c7-44fa-b8ff-67e1cf092018",
    "readOnly": false,
    "eventType": "AwsServiceEvent",
    "managementEvent": true,
    "recipientAccountId": "123456789012",
    "serviceEventDetails":
    {
        "serviceAccountId": "111122223333"
    },
    "eventCategory": "Management"
}
```

的示例 CloudTrail 事件 CreateIdentityPool

以下示例是为进行 `CreateIdentityPool` 操作而发出的请求的日志条目。该请求由名为 Alice 的 IAM 用户发出。

```
{
    "eventVersion": "1.03",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "PRINCIPAL_ID",
        "arn": "arn:aws:iam::123456789012:user/Alice",
        "accountId": "123456789012",
        "accessKeyId": "['EXAMPLE_KEY_ID']",
        "userName": "Alice"
    },
    "eventTime": "2016-01-07T02:04:30Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "CreateIdentityPool",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "127.0.0.1",
    "userAgent": "USER_AGENT",
    "requestParameters": {
        "identityPoolName": "TestPool",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "responseElements": {
        "identityPoolName": "TestPool",
        "identityPoolId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "allowUnauthenticatedIdentities": true,
        "supportedLoginProviders": {
            "graph.facebook.com": "000000000000000"
        }
    },
    "requestID": "15cc73a1-0780-460c-91e8-e12ef034e116",
    "eventID": "f1d47f93-c708-495b-bff1-cb935a6064b2",
    "eventType": "AwsApiCall",
    "recipientAccountId": "123456789012"
}
```

的示例 CloudTrail 事件 GetCredentialsForIdentity

以下示例是为进行 `GetCredentialsForIdentity` 操作而发出的请求的日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetCredentialsForIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-credentials-for-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "responseElements": {
        "credentials": {
            "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
            "sessionToken": "aAaAaAaAaAaAab1111111111EXAMPLE",
            "expiration": "Jan 19, 2023 5:55:08 PM"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "659dfc23-7c4e-4e7c-858a-1abce884d645",
    "eventID": "6ad1c766-5a41-4b28-b5ca-e223ccb00f0d",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
```

```
}
```

的示例 CloudTrail 事件 GetId

以下示例是为进行 `GetId` 操作而发出的请求的日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:05Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetId",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-id",
    "requestParameters": {
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "dc28def9-07c8-460a-a8f3-3816229e6664",
    "eventID": "c5c459d9-40ec-41fd-8f6b-57865d5a9975",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

的示例 CloudTrail 事件 GetOpenIdToken

以下示例是为进行 `GetOpenIdToken` 操作而发出的请求的日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdToken",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token",
    "requestParameters": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "a506ba18-10d7-4fdb-9548-a8187b2e38bb",
    "eventID": "19ffc1a6-6ed8-4580-a4e1-3062c5ce6457",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

的示例 CloudTrail 事件 GetOpenIdTokenForDeveloperIdentity

以下示例是为进行 `GetOpenIdTokenForDeveloperIdentity` 操作而发出的请求的日志条目。


```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "AssumedRole",
        "principalId": "AROA1EXAMPLE:johns-AssumedRoleSession",
        "arn": "arn:aws:sts::111122223333:assumed-role/Admin/johns-AssumedRoleSession",
        "accountId": "111122223333",
        "accessKeyId": "ASIAIOSFODNN7EXAMPLE",
        "sessionContext": {
            "sessionIssuer": {
                "type": "Role",
                "principalId": "AROA1EXAMPLE",
                "arn": "arn:aws:iam::111122223333:role/Admin",
                "accountId": "111122223333",
                "userName": "Admin"
            },
            "attributes": {
                "creationDate": "2023-01-19T16:53:14Z",
                "mfaAuthenticated": "false"
            }
        }
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "GetOpenIdTokenForDeveloperIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "27.0.3.154",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.get-open-id-token-for-developer-identity",
    "requestParameters": {
        "tokenDuration": 900,
        "identityPoolId": "us-east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE",
        "logins": {
            "JohnsDeveloperProvider": "HIDDEN_DUE_TO_SECURITY_REASONS"
        }
    },
    "responseElements": {
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE"
    },
    "requestID": "b807df87-57e7-4dd6-b90c-b06f46a61c21",
```

```
    "eventID": "f26fed91-3340-4d70-91ae-cdf555547b76",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

的示例 CloudTrail 事件 UnlinkIdentity

以下示例是为进行 `UnlinkIdentity` 操作而发出的请求的日志条目。

```
{
    "eventVersion": "1.08",
    "userIdentity": {
        "type": "Unknown"
    },
    "eventTime": "2023-01-19T16:55:08Z",
    "eventSource": "cognito-identity.amazonaws.com",
    "eventName": "UnlinkIdentity",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "192.0.2.4",
    "userAgent": "aws-cli/2.7.25 Python/3.9.11 Darwin/21.6.0 exe/x86_64 prompt/off
 command/cognito-identity.unlink-identity",
    "requestParameters": {
        "logins": {
            "cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa":
 "HIDDEN_DUE_TO_SECURITY_REASONS"
        },
        "identityId": "us-east-1:1cf667a2-49a6-454b-9e45-23199EXAMPLE",
        "loginsToRemove": ["cognito-idp.us-east-1.amazonaws.com/us-east-1_aaaaaaaaa"]
    },
    "responseElements": null,
    "requestID": "99c2c8e2-9c29-416f-bb17-b650a5cbada9",
    "eventID": "d8e26126-202a-43c2-b458-3f225efaedc7",
    "readOnly": false,
    "resources": [{
        "accountId": "111122223333",
```

```
        "type": "AWS::Cognito::IdentityPool",
        "ARN": "arn:aws:cognito-identity:us-east-1:111122223333:identitypool/us-
east-1:2dg778b3-50b7-565c-0f56-34200EXAMPLE"
    }],
    "eventType": "AwsApiCall",
    "managementEvent": false,
    "recipientAccountId": "111122223333",
    "eventCategory": "Data"
}
```

Amazon Cognito 的合规性验证

作为多项合规计划的一部分，第三方审计师对 Amazon Cognito 的安全与 Amazon 合规性进行评估。其中包括 SOC、PCI、FedRAMP、HIPAA 及其他。

有关特定合规计划范围内的 Amazon 服务列表，请参阅Amazon 按合规计划划分规计划划分）。有关常规信息，请参阅 Amazon 合规性计划。

您可以使用下载第三方审计报告 Amazon Artifact。有关更多信息，请参阅中的 “下载报告” Amazon Artifact。

您使用 Amazon Cognito 的合规性责任取决于您数据的敏感度、贵公司的合规性目标以及适用的法律法规。 Amazon 提供以下资源来帮助满足合规性：

- 安全性与合规性 Quick Start 指南安全性与合规性 Quick Start 指南 – 这些部署指南讨论了架构注意事项，并提供了在 Amazon上部署关注安全性和合规性的基准环境的步骤。
- HIPAA 安全与合规架构白皮书 — 本白皮书描述了公司如何使用来 Amazon 创建符合 HIPAA 标准的应用程序。
- 合规资源 — 此工作簿和指南集合可能适用于您的行业和所在地区。
- 使用《Amazon Config 开发人员指南》中的规则评估资源 — Amazon Config; 评估您的资源配置在多大程度上符合内部实践、行业准则和法规。
- Amazon Security Hub— 此 Amazon 服务可全面了解您的安全状态 Amazon ，帮助您检查是否符合安全行业标准和最佳实践。

Amazon Cognito 中的恢复能力

Amazon 全球基础设施是围绕 Amazon 区域和可用区构建的。各区域提供多个在物理上独立且隔离的可用区，这些可用区通过延迟低、吞吐量高且冗余性高的网络连接在一起。利用可用区，您可以设计和

操作在可用区之间无中断地自动实现故障转移的应用程序和数据库。与传统的单个或多个数据中心基础结构相比，可用区具有更高的可用性、容错性和可扩展性。

有关 Amazon 区域和可用区的更多信息，请参阅Amazon 全球基础设施。

主题

- 区域数据注意事项

区域数据注意事项

Amazon Cognito 用户池均在一个 Amazon 区域中创建，并且它们仅在该区域存储用户个人资料数据。用户池可以将用户数据发送到不同的 Amazon 区域，具体取决于可选功能的配置方式。

- 如果默认的 `no-reply@verificationemail.com` 电子邮件地址设置用于通过 Amazon Cognito 用户池路由电子邮件地址验证，则电子邮件将通过与关联用户池相同的区域路由。
- 如果使用不同的电子邮件地址来配置带有 Amazon Cognito 用户池的亚马逊简单电子邮件服务 (Amazon SES)，则该电子邮件地址将 Amazon 通过与 Amazon SES 中的电子邮件地址关联的区域进行路由。
- 除非在配置电子邮件或电话验证中另有说明，否则来自 Amazon Cognito 用户池的短信通过同一区域 Amazon SNS 路由。
- 如果 Amazon Pinpoint 分析用于 Amazon Cognito 用户池，则事件数据将路由到美国东部（弗吉尼亚北部）区域。

> **Note**
>
> Amazon Pinpoint 已在北美、欧洲、亚洲和大洋洲的多个 Amazon 地区上市。Amazon Pinpoint 区域包括 Amazon Pinpoint API。如果 Amazon Cognito 支持 Amazon Pinpoint 区域，Amazon Cognito 会将事件发送到同一 Amazon Pinpoint 区域中的 Amazon Pinpoint 项目。如果区域不受 Amazon Pinpoint 支持，Amazon Cognito 将仅支持在 us-east-1 中发送事件。有关 Amazon Pinpoint 区域的详细信息，请参阅 Amazon Pinpoint 端点和配额和将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用。

Amazon Cognito 中的基础设施安全性

作为一项托管服务，Amazon Cognito 受到 Amazon 全球网络安全的保护。有关 Amazon 安全服务以及如何 Amazon 保护基础设施的信息，请参阅Amazon 云安全。要使用基础设施安全的最佳实践来设计您的 Amazon 环境，请参阅 S Amazon ecurity Pillar Well-Architected Fram ework 中的基础设施保护。

您可以使用 Amazon 已发布的 API 调用通过网络访问 Amazon Cognito。客户端必须支持以下内容：

- 传输层安全性协议（TLS）。我们要求使用 TLS 1.2，建议使用 TLS 1.3。
- 具有完全向前保密（PFS）的密码套件，例如 DHE（临时 Diffie-Hellman）或 ECDHE（临时椭圆曲线 Diffie-Hellman）。大多数现代系统（如 Java 7 及更高版本）都支持这些模式。

此外，必须使用访问密钥 ID 和与 IAM 主体关联的秘密访问密钥来对请求进行签名。或者，您可以使用 Amazon Security Token Service（Amazon STS）生成临时安全凭证来对请求进行签名。

Amazon Cognito 用户池中的配置和漏洞分析

Amazon 处理基本的安全任务，例如客户机操作系统 (OS) 和数据库修补、防火墙配置和灾难恢复。这些流程已通过相应第三方审核和认证。有关更多详细信息，请参阅以下资源：

- Amazon Cognito 的合规性验证
- 责任共担模式

Amazon 亚马逊 Cognito 的托管策略

要向用户、群组和角色添加权限，使用 Amazon 托管策略比自己编写策略要容易得多。创建仅为团队提供所需权限的 IAM 客户管理型策略需要时间和专业知识。要快速入门，您可以使用我们的 Amazon 托管策略。这些政策涵盖常见用例，可在您的 Amazon 账户中使用。有关 Amazon 托管策略的更多信息，请参阅 IAM 用户指南中的Amazon 托管策略。

Amazon 服务维护和更新 Amazon 托管策略。您无法更改 Amazon 托管策略中的权限。服务偶尔会向 Amazon 托管式策略添加额外权限以支持新特征。此类更新会影响附加策略的所有身份（用户、组和角色）。当启动新特征或新操作可用时，服务最有可能会更新 Amazon 托管式策略。服务不会从 Amazon 托管策略中移除权限，因此策略更新不会破坏您的现有权限。

此外，还 Amazon 支持跨多个服务的工作职能的托管策略。例如，ReadOnlyAccess Amazon 托管策略提供对所有 Amazon 服务和资源的只读访问权限。当服务启动一项新功能时， Amazon 会为新操作和资源添加只读权限。有关工作职能策略的列表和说明，请参阅 IAM 用户指南中的适用于工作职能的 Amazon 托管式策略。

Amazon 授予亚马逊 Cognito 访问权限的托管 IAM 策略

- `AmazonCognitoPowerUser`– 用于对身份池和用户池所有方面进行访问和管理的权限。要查看此策略的权限，请参阅AmazonCognitoPowerUser。
- `AmazonCognitoReadOnly` – 用于对身份池和用户池进行只读访问的权限。要查看此策略的权限，请参阅AmazonCognitoReadOnly。
- `AmazonCognitoDeveloperAuthenticatedIdentities` – 用于将身份验证系统与 Amazon Cognito 集成的权限。要查看此策略的权限，请参阅AmazonCognitoDeveloperAuthenticatedIdentities。

这些政策由 Amazon Cognito 团队维护，因此，即使添加了新 APIs 政策，您的用户仍可继续拥有相同的访问级别。

> **Note**
>
> 创建新的身份池时，您可以自动为经过身份验证的用户和访客用户访问权限创建新角色。使用新的 IAM 角色创建身份池的管理员还必须拥有 IAM 权限才能创建角色。

具有未经身份验证的访客访问权限的身份池将额外的 Amazon 托管策略作为会话策略应用于未经身份验证的用户。此 Amazon 托管策略没有预期的管理用途。相反，它限制了您可以对身份池增强的身份验证流程中的访客用户应用的权限范围。有关更多信息，请参阅 IAM 角色。

Amazon Amazon Cognito 向访客用户授予的托管 IAM 策略

- `AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 结合内联会话策略，限制 IAM 管理员可以向身份池访客用户授予的权限。Amazon Cognito 会自动将此政策应用于访客会话。有关更多信息，请参阅 访客 Amazon 托管会话政策。

亚马逊 Cognito 更新了托管 Amazon 政策

查看自该服务开始跟踪这些更改以来，Amazon Cognito 托 Amazon 管政策更新的详细信息。有关此页面更改的自动提示，请订阅 Amazon Cognito 文档历史记录页面上的 RSS 源。

更改	描述	日期
`AmazonCognitoPowerUser` - 更改	Amazon Cognito 添加了新操作，允许亚马逊 Cognito 用户池管理高级用户使用 Amazon End User Messaging SMS API 操作DescribeAccountAttributes。	2025 年 2 月 27 日
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 更改	Amazon Cognito 添加了新的操作，允许身份池 Amazon Key Management Service 中未经身份验证的（访客）用户使用。	2024 年 10 月 30 日
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 更改	Amazon Cognito 添加了新的操作，允许身份池中未经身份验证的（访客）用户使用 Amazon Location Service。	2024 年 8 月 9 日
`AmazonCognitoUnAuthedIdentitiesSessionPolicy` - 新策略	添加了用于缩小身份池中访客用户的权限范围的 Amazon 托管策略。	2023 年 7 月 14 日
`AmazonCognitoPowerUser` 和 `AmazonCognitoReadOnly` - 更改	添加了新的权限，允许高级用户查看和管理 Amazon WAF 网站 ACLs 与 Amazon Cognito 用户池的关联。 添加了新的权限，允许只读用户查看 Amazon WAF 网站	2022 年 7 月 19 日

更改	描述	日期
	ACLs 与 Amazon Cognito 用户池的关联。	
`AmazonCognitoPowerUser` - 更改	添加了一项新权限，允许 Amazon Cognito 调用 Amazon Simple Notification Service `PutIdentityPolicy` 和 `ListConfigurationSets` 操作。 此更改允许 Amazon Cognito 用户池更新 Amazon SES 发送授权策略，并在您配置用户池中发送电子邮件时应用 Amazon SES 配置集。	2021 年 11 月 17 日
`AmazonCognitoPowerUser` - 更改	添加了一项新权限，允许 Amazon Cognito 调用 Amazon Simple Notification Service 的 `GetSMSSandboxAccountStatus` 操作。 此更改允许 Amazon Cognito 用户池决定您是否需要退出 Amazon Simple Notification Service 沙盒，以便通过用户池向所有终端用户发送消息。	2021 年 6 月 1 日
Amazon Cognito 开启跟踪更改	Amazon Cognito 开始跟踪其托 Amazon 管策略的变更。	2021 年 3 月 1 日

为 Amazon Cognito 资源贴标签

标签是您或 Amazon 分配给 Amazon 资源的元数据标签。每个标签均包含一个键 和一个值。对于您分配的标签，需要定义键和值。例如，您可以将键定义为 `stage`，将一个资源的值定义为 `test`。

标签可帮助您：

- 识别和整理您的 Amazon 资源。许多 Amazon 服务都支持标记，因此您可以为来自不同服务的资源分配相同的标签。这有助于您指示哪些资源是相关的。例如，您可以将相同的标签分配给您分配至 Amazon DynamoDB 表的 Amazon Cognito 用户池。
- 追踪您的 Amazon 成本。您可以在 Amazon 账单与成本管理 控制面板上激活这些标签。 Amazon 使用成本分配标签对您的成本进行分类，并向您提供每月成本分配报告。有关更多信息，请参阅《Amazon Billing 用户指南》中的使用成本分配标签。
- 根据分配给资源的标签控制对资源的访问。您可以通过在 Amazon Identity and Access Management（IAM）策略的条件中指定标签键和值来控制访问权限。例如，您可以允许用户更新用户群体，但前提是该用户群体具有 `owner` 标签且值为该用户的名称。有关更多信息，请参阅《IAM 用户指南》中的使用标签控制访问。

您可以使用 Amazon Command Line Interface 或 Amazon Cognito API 为用户和身份池添加、编辑或删除标签。您还可以使用 Amazon Cognito 控制台来管理用户池的标签。

有关使用标签的提示，请参阅 Amazon Answers 博客上的 Amazon tagging strategies 文章。

以下各部分提供有关 Amazon Cognito 标签的更多信息。

Amazon Cognito 中支持的资源

Amazon Cognito 中的以下资源支持贴标签：

- 用户池
- 身份池

标签限制

以下限制适用于 Amazon Cognito 资源上的标签：

- 您可以分配给资源的最大标签数量 - 50

- 最大密钥长度 – 128 个 Unicode 字符
- 最大值长度 – 256 个 Unicode 字符
- 键和值的有效字符 – a-z、A-Z、0-9、空格和以下字符：_ . : / = + - 和 @
- 键和值区分大小写
- 请不要使用 `aws:` 作为键的前缀；它保留为供 Amazon 使用

使用 Amazon Cognito 控制台管理标签

您可以使用 Amazon Cognito 控制台来管理分配给您的用户池的标签。

将标签添加到用户池

1. 导航到 Amazon Cognito 控制台。如果出现提示，请输入您的 Amazon 凭据。
2. 选择用户池。
3. 从列表中选择一个现有用户池，或创建一个用户池。
4. 选择 “设置” 菜单并找到 “标签” 选项卡。
5. 选择 Add tags（添加标签）即可添加您的第一个标签。如果您之前已将标签分配给此用户池，请在 Manage tags（管理标签）中选择 Add another（添加其它）。
6. 为 Tag key（标签键）和 Tag value（标签值）指定值。
7. 对于每个要添加的其它标签，请选择 Add another（添加其它）。
8. 添加完标签后，选择 Save changes（保存更改）。

要标记身份池，请导航至身份池菜单，然后选择或创建身份池。在 “身份池属性” 选项卡中，找到 “标签”。选择 Add tag（添加标签）。

Amazon CLI 例子

Amazon CLI 提供的命令可帮助您管理分配给 Amazon Cognito 用户池和身份池的标签。

分配标签

使用以下命令可将标签分配给现有的用户池和身份池。

Example 适用于用户池的 **`tag-resource`** 命令

使用 `cognito-idp` 命令集中的 `tag-resource`将标签分配给用户池：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test
```

此命令包含以下参数：

- `resource-arn` - 您向其应用标签的用户池的 Amazon Resource Name（ARN）。要查找 ARN，请在 Amazon Cognito 控制台中选择该用户池，并查看 General settings（常规设置）选项卡上的 Pool ARN（池 ARN）值。
- `tags` – 以格式 *`key`*=*`value`* 表示的标签的键值对。

要一次分配多个标签，请以逗号分隔的列表形式指定它们：

```
$ aws cognito-idp tag-resource \
> --resource-arn user-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

Example 适用于身份池的 **`tag-resource`** 命令

使用 `cognito-identity` 命令集中的 `tag-resource` 将标签分配给身份池：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test
```

此命令包含以下参数：

- `resource-arn` - 您向其应用标签的身份池的 Amazon Resource Name（ARN）。要查找 ARN，请在 Amazon Cognito 控制台中选择身份池，并选择 Edit identity pool (编辑身份池)。然后，在 Identity pool ID（身份池 ID）中，选择 Show ARN（显示 ARN）。
- `tags` – 以格式 *`key`*=*`value`* 表示的标签的键值对。

要一次分配多个标签，请以逗号分隔的列表形式指定它们：

```
$ aws cognito-identity tag-resource \
> --resource-arn identity-pool-arn \
> --tags Stage=Test,CostCenter=80432,Owner=SysEng
```

查看标签

使用以下命令可查看您已分配给用户池和身份池的标签。

Example 适用于用户池的 **list-tags-for-resource** 命令

使用 cognito-idp 命令集中的 list-tags-for-resource 查看分配给用户池的标签：

```
$ aws cognito-idp list-tags-for-resource --resource-arn user-pool-arn
```

Example 适用于身份池的 **list-tags-for-resource** 命令

使用 cognito-identity 命令集中的 list-tags-for-resource 查看分配给身份池的标签：

```
$ aws cognito-identity list-tags-for-resource --resource-arn identity-pool-arn
```

删除标签

使用以下命令从用户池和身份池中删除标签。

Example 适用于用户池的 **untag-resource** 命令

使用 cognito-idp 命令集中的 untag-resource 删除用户池中的标签：

```
$ aws cognito-idp untag-resource \
> --resource-arn user-pool-arn \
> --tag-keys Stage CostCenter Owner
```

对于 --tag-keys 参数，请指定一个或多个标签键。请勿包含标签值。用空格分隔键。

Example 适用于身份池的 **untag-resource** 命令

使用 cognito-identity 命令集中的 untag-resource 删除身份池中的标签：

```
$ aws cognito-identity untag-resource \
> --resource-arn identity-pool-arn \
> --tag-keys Stage CostCenter Owner
```

对于 --tag-keys 参数，请指定一个或多个标签键。请勿包含标签值。

> **Important**
>
> 删除用户或身份池后，与已删除池相关的标签在删除后的 30天内仍然可以在控制台或 API 调用中显示。

在创建资源时应用标签

使用以下命令可在创建用户池或身份池时分配标签。

Example **`create-user-pool`** 命令以及标签

当您使用 `create-user-pool` 命令创建用户时，您可以使用 `--user-pool-tags` 参数指定标签：

```
$ aws cognito-idp create-user-pool \
> --pool-name user-pool-name \
> --user-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

标签的键值对必须采用格式 *`key`*=*`value`*。如果要添加多个标签，请以逗号分隔的列表形式指定它们。

Example **`create-identity-pool`** 命令以及标签

当您使用 `create-identity-pool` 命令创建用户池时，您可以使用 `--identity-pool-tags` 参数指定标签：

```
$ aws cognito-identity create-identity-pool \
> --identity-pool-name identity-pool-name \
> --allow-unauthenticated-identities \
> --identity-pool-tags Stage=Test,CostCenter=80432,Owner=SysEng
```

标签的键值对必须采用格式 *`key`*=*`value`*。如果要添加多个标签，请以逗号分隔的列表形式指定它们。

使用 Amazon Cognito API 管理标签

您可以在 Amazon Cognito API 中使用以下操作来管理用户池和身份池的标签。

适用于用户池标签的 API 操作

使用以下 API 操作来分配、查看和删除用户池的标签。

- TagResource
- ListTagsForResource
- UntagResource
- CreateUserPool

适用于身份池标签的 API 操作

使用以下 API 操作来分配、查看和删除身份池的标签。

- TagResource
- ListTagsForResource
- UntagResource
- CreateIdentityPool

Amazon Cognito 中的限额

Amazon Cognito 对您可以在账户中执行的最大操作数具有默认限额，以前称为限制。Amazon Cognito 对于 Amazon Cognito 资源的最大数量和规模也具有限额。

每个 Amazon Cognito 配额将最大请求量合而为一 Amazon Web Services 区域 。 Amazon Web Services 账户例如，您的应用程序可以对美国东部（弗吉尼亚州北部）中的所有用户池，最高以默认配额（RPS）速率为 `UserAuthentication` 操作发出 API 请求。您在亚太地区（东京）的应用程序可以针对您所在地区的所有用户池生成相同数量的请求。 Amazon 一次只能在一个地区批准增加配额的请求。在美国东部（弗吉尼亚州北部）成功增加限额对您在亚太地区（东京）的最大请求速率没有任何效果。

主题

- 了解 API 请求速率配额
- 管理 API 请求速率配额
- Amazon Cognito 用户池 API 操作类别和请求速率配额
- Amazon Cognito 身份池（联合身份）API 操作请求速率配额
- 资源数量和大小的配额

了解 API 请求速率配额

配额分类

Amazon Cognito 为 API 操作强制执行最大请求速率。有关 Amazon Cognito 提供的 API 操作的更多信息，请参阅《API 参考指南》中的用户池和身份池。对于用户池，这些操作分为多个常见使用场景类别，例如 `UserAuthentication` 或 `UserCreation`。有关按类别列出的用户池 API 操作列表，请参阅Amazon Cognito 用户池 API 操作类别和请求速率配额。

在 Service Quotas 控制台中，您可以按用户池和身份池类别跟踪配额使用情况。如果您的 Amazon Cognito 用户池的请求速率超过配额，则可以购买更多容量。在 Service Quotas 控制台中，您可以按类别跟踪用户池配额使用情况以及购买增加的配额。

操作限额定义如下：某类别内所有操作的每秒最大请求数（RPS）。Amazon Cognito 用户池服务将配额应用于每个类别下的所有操作。例如，类别 `UserCreation`，包括四项操作：`SignUp`、`ConfirmSignUp`、`AdminCreateUser` 和 `AdminConfirmSignUp`。该类别分配有组

合配额 50 RPS。如果多个操作同时发生，此类别中的每个操作最多可以单独调用 50 RPS，也可以组合调用。

> **Note**
>
> 类别配额仅适用于用户池。Amazon Cognito 将每个身份池配额应用于单个操作。对于每个类别和每个操作的请求速率配额， Amazon 衡量来自一个区域内所有用户池或身份池的所有请求的总速率。 Amazon Web Services 账户

使用特殊请求速率处理 Amazon Cognito 用户池 API 操作

操作配额针对组合请求总数在类别级别进行衡量和实施，但 `AdminRespondToAuthChallenge` 和 `RespondToAuthChallenge` 操作除外，在这两个操作中，将应用特殊处理规则。

`UserAuthentication` 类别包括 Amazon Cognito 用户池 API 中的四个操作：`AdminInitiateAuth`、`InitiateAuth`、`AdminRespondToAuthChallenge` 和 `RespondToAuthChallenge`。此外，托管 UI 中的用户身份验证也会计入这个配额。`InitiateAuth` 和 `AdminInitiateAuth` 操作按照类别配额进行衡量和执行。匹配的 `RespondToAuthChallenge` 和 `AdminRespondToAuthChallenge` 操作需依据单独的配额，该配额是 `UserAuthentication` 类别限制的三倍。这个提升的配额适合您的应用程序中设置的多重身份验证质询。此配额足以涵盖大多数使用案例。在您的应用程序对身份验证质询作出最多三次响应后，其他请求将计入 `UserAuthentication` 类别配额。多重身份验证（MFA）、设备身份验证和自定义身份验证都是您可以设计到用户池中的质询提示示例。

例如，如果 `UserAuthentication` 类别的配额为 80 RPS，您可以按照高达 240 RPS（3 * 80 RPS）的速率调用 `RespondToAuthChallenge` 或 `AdminRespondToAuthChallenge`。如果您的用户池每次身份验证提示进行四轮质询，并且每秒有 70 个用户登录，则总 `RespondToAuthChallenge` 为 280 RPS（70 x 4），比配额高出 40 RPS。将额外的 40 RPS 添加到 70 个 `InitiateAuth` 调用，使 `UserAuthentication` 类别的总使用量为 110 RPS（40 + 70）。由于此值比设置为 80 RPS 的类别配额高出 30 RPS，因此 Amazon Cognito 会限制来自您的应用程序的请求。

每月活跃用户

当 Amazon Cognito 计算用户池账单时，它会根据每个每月活跃用户（MAU）来收费。在为配额增加请求做计划时，请考虑您当前和预计的 MAU 数量。如果在一个日历月内有与该用户相关的身份操作，则该用户将计为 MAU。当您将联合用户与本地用户关联时，MAU 计数为一加 n，其中 n 是已登录的关联身份数。使用户变为活动状态的活动包括以下各项。

- 注册或管理创建用户。用户 CSV 导入不会增加您的 MAU 数量。
- 用户账户确认或属性验证。
- 登录和挑战回复。您使用当前登录用户的访问令牌授权的操作不会影响您的 MAU 计数；但是，由于登录会生成访问令牌，因此这些操作表明关联用户是 MAU。
- 注销和令牌撤销。
- 以管理员身份自助重置密码和设置用户密码。以管理员身份@@ 重置用户密码 (AdminResetUserPassword) 不会增加您的 MAU 数量。
- 更改用户属性或群组成员资格。
- 以管理员身份查询用户的详细属性。

> **Note**
>
> “以管理员身份查询用户的详细属性” 类别包括 API 操作 AdminGetUser，但不包括ListUsers。大型用户池中的详细 user-by-user查询可能会对您的 Amazon 账单产生重大影响。为避免额外成本，请使用外部数据库收集用户数据`ListUsers`或将用户信息存储在外部数据库中。

任何活跃用户或在一个日历月内未处于活动状态的任何用户都不会向您收取额外会话费用。在您在 Lite、E ssent ial s 和 Plus 的可用选项之间更改了用户池功能套餐的月份中，该月的账单是根据每个等级的每月活跃用户总数 (MAUs) 计算的，当用户处于活动状态时，每个 MAU 都分配到价格最高的分配级别。例如：

1. 在月初，您的用户群已加入 Plus 功能套餐。
2. 用户 A 在当月的第一天登录。
3. 用户 B 在当月的第一天和最后一天登录。
4. 在每月的第十天，您将功能计划切换到基本套餐。
5. 用户 C 在该月的最后一天登录。

在这种情况下，用户 A 和用户 B 是 Plus MAUs ，用户 C 是 Essentials MAU。

精简版 MAU

当用户池使用精简版功能计划时，每月至少活跃一次，而当用户池使用精简版或增强版计划时，该用户从未处于活动状态。

必需品 MAU

在用户池加入 Essentials 功能计划时，每月至少处于活动状态一次的用户，而在用户池加入增强版计划时从未处于活动状态。

再加上 MAU

在用户池加入 Plus 套餐时，每月至少活跃一次的用户。

有关更多信息，请参阅 用户池功能计划。

管理 API 请求速率配额

确定配额要求

> **Important**
>
> 如果您增加 `UserAuthentication`、`UserCreation` 或 `AccountRecovery` 等类别的 Amazon Cognito 配额，您可能需要增加其他 Amazon Web Services 服务的配额。例如，如果这些服务的请求率配额不足，Amazon Cognito 使用 Amazon Simple Notification Service（Amazon SNS）或 Amazon Simple Email Service（Amazon SES）发送的邮件可能会发送失败。

要计算配额要求，请确定在特定时间段内将与您的应用程序交互的活跃用户数。例如，如果您的应用程序预计在八小时内平均有 100 万个活跃用户登录，则您必须平均每秒对 35 个用户进行身份验证。

此外，如果您假设平均用户会话为两个小时，并且令牌配置为在一个小时后过期，则每个用户必须在此会话期间刷新其令牌一次。为支持此负载，`UserAuthentication` 类别所需的平均配额为 70 RPS。

如果考虑到八小时内用户登录频 peak-to-average率的差异，假设比率为 3:1，则需要所需的`UserAuthentication`配额 200 RPS。

> **Note**
>
> 如果您为每个用户操作调用多个操作，则必须在类别级别对各个操作调用速率进行求和。

优化请求速率以避免达到配额限制

由于提高 API 速率限制会增加 Amazon 账单成本，因此在申请增加配额之前，请考虑调整使用模式。以下是优化请求速率的一些应用程序架构示例。

在回退等待期之后重试尝试

> 您可以在每次 API 调用时捕获错误，然后在回退期之后重试尝试。您可以根据业务需求和负载调整回退算法。Amazon 内置 SDKs 了重试逻辑。有关更多信息，请参阅构建工具 Amazon。

使用外部数据库处理频繁更新的属性

> 如果您的应用程序需要对用户池进行多次调用以读取或写入自定义属性，可使用外部存储。您可以使用首选数据库存储自定义属性，也可以在登录期间使用缓存层来加载用户配置文件。您可以在需要时从缓存中引用此配置文件，而无需从用户池重新加载用户配置文件。

在客户端验证 JSON 网络令牌 (JWTs)

> 应用程序必须在信任 JWT 令牌之前验证这些令牌。您可以在客户端验证令牌的签名和有效性，而无需向用户池发送 API 请求。验证令牌后，您可以信任令牌中的陈述并使用陈述，而不是执行更多 `getUser` API 调用。有关更多信息，请参阅 验证 JSON Web 令牌。

使用等候室限制 Web 应用程序的流量

> 如果您预计在有时限的活动（例如参加考试或参加实时活动）期间会有大量用户登录，则可以使用自我限制机制优化请求流量。例如，您可以设置一个等候室，用户可以在其中等待直到会话可用，从而允许您在有可用容量时处理请求。请参阅 Amazon 虚拟等候室解决方案以获取等候室的参考架构。

缓存 JWTs

> 在访问令牌过期前重用令牌。有关在 API Gateway 中使用令牌缓存的框架示例，请参阅管理用户池令牌到期和缓存。与其生成 API 请求来查询用户信息，不如在 ID 令牌到期前先缓存令牌，然后从缓存中读取用户属性。

有关在中使用 API 请求速率的更多信息 Amazon，请参阅管理和监控工作负载中的 API 限制。有关优化会增加 Amazon 账单成本的 Amazon Cognito 操作的信息，请参阅。管理 成本

跟踪配额使用量

Amazon Cognito 在亚马逊中 CloudWatch 为账户级别的每个 API 操作类别生成`CallCountThrottleCount`指标。您可以使用 `CallCount` 跟踪客户发出的与类别相关的调用

总数。您可以使用 `ThrottleCount` 跟踪与类别相关的节流调用总数。您可以使用 `CallCount` 和 `ThrottleCount` 指标以及 `Sum` 统计数据计算类别中的调用总数。有关更多信息，请参阅CloudWatch 使用量指标。

监控服务配额时，利用率是使用中的服务配额的百分比。例如，如果配额值为 200 个资源，正在使用 150 个资源，则利用率为 75%。使用量是指服务配额中正在使用的资源或操作的数量。

通过 CloudWatch 指标跟踪使用情况

您可以使用跟踪和收集 Amazon Cognito 用户池使用率指标。 CloudWatch CloudWatch仪表板会显示有关您 Amazon Web Services 服务 使用的每一项的指标。使用 CloudWatch，您可以创建指标警报来通知您或更改您正在监控的特定资源。有关 CloudWatch 指标的更多信息，请参阅跟踪您的 CloudWatch 使用量指标。

通过 Service Quotas 指标跟踪利用率

Amazon Cognito 用户池已与 Service Quotas 集成，Service Quotas 是一个控制台界面，可用于显示和管理您的服务配额使用量。在 Service Quotas 控制台中，您可以查找特定配额的值、查看监控信息、请求增加配额或设置 CloudWatch 警报。在您的账户处于活动状态一段时间后，您可以查看资源利用率图。

Service Quotas 控制台中适用于 Amazon Cognito 用户池和 Amazon Cognito 身份池的已应用的账户级别配额值列显示您的当前配额。利用率列显示您当前的配额使用率。可调整的 Amazon Cognito 用户池 requests-per-second (RPS) 配额会显示其当前使用情况。Service Quotas 控制台还可以将您导航到 CloudWatch 指标，以便仔细查看所选配额指标。有关在 Service Quotas 控制台中查看配额的更多信息，请参阅查看 Service Quotas。

跟踪每月活跃用户 (MAUs)

用户池中的每月活跃用户数 (MAUs) 为你计划增加请求率配额提供了重要数据。您可以将您的 API 请求速率与您在给定时间段内的活跃用户数量进行比较。有了这些信息，您就可以计算出应用程序活跃用户的增加将如何影响使用模型中的配额。例如，假设您在美国西部（俄勒冈州）的合并应用程序在一个月内产生了 200 万个活跃用户，而您的 `UserAuthentication` 类别在默认的每秒 120 次请求（RPS）配额下，偶尔出现了限流错误。在广告活动成功之前的上个月，你有100万个， MAUs而且你的申请从未超过80个RPS。如果您预计新的电视广告会导致出现类似的流量激增，则可能需要额外购买 40 RPS 的配额，使调整后的配额为 160 RPS，从而可容纳下一个百万用户。

要查看您的 MAUs

访问 Amazon Billing 控制台并查看最近的账单。在 “按服务计费” 下，您可以在 Cognito 上进行筛选，以查看该账单周期的明细。 MAUs

请求提高限额

Amazon Cognito 对您在用户池和每个用户池中的身份池中每秒可以执行的最大操作数都有配额。Amazon Web Services 区域您可以购买额外的配额来增加可调整 Amazon Cognito 用户池 API 请求速率配额。查看您当前的配额，然后通过 Service Quotas 控制台或通过 Service Quotas API 操作 `ListAWSDefaultServiceQuotas` 和 `RequestServiceQuotaIncrease` 购买增量配额。

- 要使用 Service Quotas 控制台购买增量配额，请参阅《Service Quotas 用户指南》中的请求增加 API 配额。
- Amazon 目标是在 10 天内完成增加配额的请求。然而，一些因素可能导致请求处理时间超过 10 天。例如，某些请求可能要求 Amazon Cognito 预配置额外的硬件容量，而请求量的季节性增加可能会导致延迟。
- 如果配额在 Service Quotas 中尚不可用，请使用服务限制提高表单。

> **Important**
>
> 只能增加可调配额。您必须购买增加的配额容量。有关提高配额的定价，请参阅 Amazon Cognito 定价。

Amazon Cognito 用户池 API 操作类别和请求速率配额

由于 Amazon Cognito 的重叠 API 操作类具有不同的授权模型，因此每个操作都属于一个类别。每个类别对所有成员 API 操作都有自己的共同使用配额，跨您账户中一个 Amazon Web Services 区域 的所有用户池。您只能请求提升可调整类别配额。有关更多信息，请参阅 请求提高限额。配额调整适用于您的账户在单个区域中的用户池。对于每个用户池，Amazon Cognito 将某些类别[3]中的操作限制为每秒 5 次请求（RPS）。默认配额（RPS）还适用于 Amazon Web Services 账户中的所有用户池。

> **Note**
>
> 每个类别的配额以每月活跃用户数 (MAUs) 来衡量。 Amazon Web Services 账户 少于 200 万 MAUs 可以在默认配额内运行。如果您的应用程序少于一百万， MAUs 而且 Amazon Cognito 正在限制请求，请考虑优化您的应用程序。有关更多信息，请参阅 优化请求速率以避免达到配额限制。

类别操作配额适用于一个 Amazon Web Services 区域的所有用户池中的所有用户。Amazon Cognito 还为您的应用程序可以针对一个用户生成的请求数量保留了配额。您必须限制每个用户的 API 请求数，如下表所示。

Amazon Cognito 用户池中每个用户的请求速率配额

操作	每个用户每秒操作数
读取用户配置文件 示例：`GetUser`、`GetDevice` 、`InitiateAuth` 、`RespondToAuthChallenge` 。	10
写入用户配置文件 示例：`UpdateUserAttributes` 、`SetUserSettings`	10

您必须限制每个类别的 API 请求数，如下表所示。

Amazon Cognito 用户池中每个类别的请求速率配额

类别	描述	默认配额（RPS）	可调整
`UserAuthentication` • InitiateAuth • 使用 `InitiateAuth` 或 令牌端点 刷新令牌 • RespondToAuthChallenge[1] • AdminInitiateAuth • AdminRespondToAuthChallenge[1]	对用户进行身份验证（登录）的操作。 这些操作受使用特殊请求速率处理 Amazon Cognito 用户池 API 操作 的制约。	120	是

类别	描述	默认配额（RPS）	可调整
• 托管 UI 登录和授权码或隐式授权中的 MFA[2]			
`UserCreation` • SignUp • ConfirmSignUp • AdminCreateUser • AdminConfirmSignUp	用于创建或确认 Amazon Cognito 本地用户的操作。这是由您的 Amazon Cognito 用户池直接创建和验证的用户。	50	可以
`UserFederation` 通过第三方身份提供商将用户联合（身份验证）到您的 Amazon Cognito 用户池的操作。	向用户池联合身份验证端点提交 IdP 响应的操作。生成 IdP 令牌的 OIDC 或社交服务提供商操作以及所有 SAML 请求一起构成此限额。	25	是

类别	描述	默认配额（RPS）	可调整
`UserAccountRecovery` • ChangePassword • ConfirmForgotPassword • ForgotPassword • AdminResetUserPassword • AdminSetUserPassword • RespondToAuthChallenge[1] • AdminRespondToAuthChallenge[1] • 托管登录密码重置	恢复用户账户或者更改或更新用户密码的操作。	30	否
`UserRead` • AdminGetUser • GetUser	从用户池中检索用户的操作。	120	是

类别	描述	默认配额（RPS）	可调整
`UserUpdate` • AdminAddUserToGroup • AdminDeleteUserAttributes • AdminUpdateUserAttributes • AdminDeleteUser • AdminDisableUser • AdminEnableUser • AdminLinkProviderForUser • AdminDisableProviderForUser • VerifyUserAttribute • DeleteUser • DeleteUserAttributes • UpdateUserAttributes • AdminUserGlobalSignOut • GlobalSignOut • AdminRemoveUserFromGroup	用于管理用户和用户属性的操作。	25	否
`UserToken` • RevokeToken	令牌管理的操作	120	是

类别	描述	默认配额（RPS）	可调整
`UserResourceRead` • AdminGetDevice • AdminListGroupsForUser • AdminListDevices • GetDevice • ListDevices • GetUserAttributeVerificationCode • ResendConfirmationCode • AdminListUserAuthEvents	从 Amazon Cognito 检索用户资源信息（如记忆设备或组成员资格）的操作。	50	可以

类别	描述	默认配额（RPS）	可调整
`UserResourceUpdate` • AdminForgetDevice • AdminUpdateAuthEventFeedback • AdminSetUserMFAPreference • AdminSetUserSettings • AdminUpdateDeviceStatus • UpdateDeviceStatus • UpdateAuthEventFeedback • ConfirmDevice • SetUserMFAPreference • SetUserSettings • VerifySoftwareToken • AssociateSoftwareToken • ForgetDevice	用于更新用户的资源信息（如记忆设备或组成员资格）的操作。	25	否
`UserList` • ListUsers • ListUsersInGroup	返回用户列表的操作。	30	否

类别	描述	默认配额（RPS）	可调整
`UserPoolRead` • DescribeUserPool • ListUserPools	读取用户池的操作。	15	否
`UserPoolUpdate` • CreateUserPool • UpdateUserPool • DeleteUserPool	创建、更新或删除用户池的操作。	15	否

类别	描述	默认配额（RPS）	可调整
UserPoolResourceRead • DescribeIdentityProvider • DescribeResourceServer • DescribeUserImportJob • DescribeUserPoolDomain • 获取 CSVHeader • GetGroup • GetSigningCertificate • GetIdentityProviderByIdentifier • GetUserPoolMfaConfig • ListGroups • ListIdentityProviders • ListResourceServers • ListTagsForResource • ListUserImportJobs • DescribeRiskConfiguration • 获取 UICustomization	从用户池中检索有关资源信息（如组或资源服务器）的操作。[3]	20	否

类别	描述	默认配额（RPS）	可调整
`UserPoolResourceUpdate` • AddCustomAttributes • CreateGroup • CreateIdentityProvider • CreateResourceServer • CreateUserImportJob • CreateUserPoolDomain • DeleteGroup • DeleteIdentityProvider • DeleteResourceServer • DeleteUserPoolDomain • SetUserPoolMfaConfig • StartUserImportJob • StopUserImportJob • UpdateGroup • UpdateIdentityProvider • UpdateResourceServer • UpdateUserPoolDomain	修改用户池中的资源（如组或资源服务器）的操作。[3]	15	否

类别	描述	默认配额（RPS）	可调整
• SetRiskConfiguration • Set UICustomization • TagResource • UntagResource			
`UserPoolClientRead` • DescribeUserPoolClient • ListUserPoolClients	检索用户池客户端的相关信息的操作。[3]	15	否
`UserPoolClientUpdate` • CreateUserPoolClient • DeleteUserPoolClient • UpdateUserPoolClient	创建、更新和删除用户池客户端的操作。[3]	15	否
`ClientAuthentication` `client_credentials` 向令牌端点发出授权类型请求。	生成用于授权 machine-to-machine 请求的凭证的操作	150	否

[1] `ChallengeName` 为 `NEW_PASSWORD_REQUIRED` 的 `RespondToAuthChallenge` 或 `AdminRespondToAuthChallenge` 响应计入 `UserAccountRecovery` 类别。所有其他质询响应计入 `UserAuthentication` 类别。

[2] 登录期间的每个托管 UI 操作都会为配额贡献一个请求。例如，登录并提供 MFA 代码的用户贡献 2 个请求。授权码授予中的令牌兑换会受到额外配额分配的限制，该配额分配的速率与您在 `UserAuthentication` 类别中的配额速率相同。

[3] 此类别中的任何单独操作都有一项约束，阻止单个用户池以高于 5 RPS 的速率调用操作。

Amazon Cognito 身份池（联合身份）API 操作请求速率配额

操作	描述	默认配额（RPS）[1]	可调整	提高配额的资格
`GetId`	检索身份池的身份 ID。	25	是	请联系您的账户团队。
`GetOpenIdToken`	在经典工作流中从身份池中检索 OpenID 令牌。	200	是	请联系您的账户团队。
`GetCredentialsForIdentity`	在增强的工作流程中从 Amazon 身份池中检索证书。	200	是	请联系您的账户团队。
`GetOpenIdTokenForDeveloperIdentity`	从开发人员工作流中的身份池中检索 OpenID 令牌。	50	可以	请联系您的账户团队。
`ListIdentities`	检索身份池 IDs 中的身份列表。	5	是	请联系您的账户团队。
`DeleteIdentities`	从身份池中删除一个或多个注册的身份。	10	是	请联系您的账户团队。
`TagResource`	将标签应用于身份池。	5	是	请联系您的账户团队。

操作	描述	默认配额（RPS）[1]	可调整	提高配额的资格
`UntagResource`	从身份池中移除标签。	5	是	请联系您的账户团队。
`ListTagsForResource`	显示应用于身份池的标签列表。	10	是	请联系您的账户团队。

[1] 默认配额是您的任意 Amazon Web Services 区域 身份池中身份池的最低请求速率配额 Amazon Web Services 账户。在某些区域，您的 RPS 配额可能会更高。

资源数量和大小的配额

资源配额是 Amazon Cognito 中资源、输入字段、持续时间和其他杂项特征的最大数量或大小。

您可以在 Service Quotas 控制台中或者通过提高服务限制表调整某些资源配额。要通过 Service Quotas 控制台请求增加配额，请参阅《Service Quotas 用户指南》中的请求增加配额。如果配额在 Service Quotas 中尚不可用，请使用服务限制提高表单。

> **Note**
>
> Amazon Web Services 账户 级别的资源配额（如每个区域的用户池）适用于每个区域中的 Amazon Cognito 资源。 Amazon Web Services 区域例如，您在美国东部（弗吉尼亚州北部）中有 1000 个用户池，在欧洲地区（斯德哥尔摩）另外有 1000 个。

下表显示了默认资源配额以及它们是否可调整。

Amazon Cognito 用户池资源配额

资源	限额	可调整	最大配额
每个用户池的应用程序端	1000	是	10000
每个区域的用户池数	1000	是	10000

资源	限额	可调整	最大配额
每个用户池的身份提供商	300	是	1000
每个用户池的资源服务器	25	是	300
每个用户池的用户数	40,000,000	是	请联系您的账户团队。
令牌生成前 Lambda 触发器中的合并更改总数[1]	5000	是	请联系您的账户团队。
每个用户池的托管登录品牌样式	10	否	不适用
每个用户池的自定义属性	50	不可以	不适用
每个属性的字符数	2,048 字节	否	不适用
自定义属性名称的字符数	20	否	不适用
密码策略中必需的最少密码字符	6–99	否	不适用
每人每天发送的电子邮件 Amazon Web Services 账户[2]	50	不可以	不适用
电子邮件主题中的字符数	140	否	不适用
电子邮件消息中的字符数	20000	否	不适用

资源	限额	可调整	最大配额
SMS 验证消息中的字符数	140	否	不适用
密码中的字符数	256	否	不适用
身份提供商名称的字符数	32	否	不适用
SAML 响应中的字符	100000	否	不适用
每个身份提供商的标识符数	50	不可以	不适用
链接到用户的身份数	5	否	不适用
每位用户的密码/WebAuthn 身份验证器	20	否	不适用
URLs 每个应用程序客户端的回调	100	否	不适用
URLs 每个应用程序客户端的注销	100	否	不适用
每个资源服务器的范围	100	否	不适用
每个应用程序客户端的范围	50	不可以	不适用
每个账户的自定义域	4	否	不适用
每个用户都可以属于的组	100	否	不适用
每个用户池的组数	10000	否	不适用

[1] 来自 令牌生成前 Lambda 触发器 的令牌可能会遇到此限额。一个事务中的访问令牌和身份令牌中现有和已添加的声明数量以及范围数量加起来必须小于或等于此配额。被隐藏的声明和范围不计入此限额。

[2] 仅当您使用 Amazon Cognito 用户池的默认电子邮件功能时，此限额才适用。要提高电子邮件发送量，请配置您的用户池以使用 Amazon SES 电子邮件配置。此限制每天在世界标准时间 09:00 重置。有关更多信息，请参阅 Amazon Cognito 用户池的电子邮件设置。

Amazon Cognito 用户池会话验证参数

令牌	配额
ID 令牌	5 分钟 – 1 天
刷新令牌	1 小时 – 3650 天
访问令牌	5 分钟 – 1 天
托管 UI 会话 Cookie	1 小时
身份验证会话令牌	3 分钟 – 15 分钟

Amazon Cognito 用户池代码安全资源限额（不可调整）

资源	限额
注册确认码有效期	24 小时
用户属性验证码有效期	24 小时
多重身份验证（MFA）代码有效期	3–15 分钟
忘记密码代码有效期	1 小时
每位用户每小时的最大 `ConfirmForgotPassword` 和 `ForgotPassword` 请求数[1]	5–20

资源	限额
每位用户每小时的最大 ResendConfirmationCode 请求数	5
每位用户每小时的最大 ConfirmSignUp 请求数	15
每位用户每小时的最大 ChangePassword 请求数	5
每位用户每小时的最大 GetUserAttributeVerificationCode 请求数	5
每位用户每小时的最大 VerifyUserAttribute 请求数	15

[1] Amazon Cognito 会评估更新密码请求中的风险因素，然后分配一个与评估的风险等级相关的配额。有关更多信息，请参阅 忘记密码行为。

Amazon Cognito 用户池用户导入任务资源限额

资源	限额	可调整	最大配额
每个用户池的用户导入任务	1000	是	请联系您的账户团队。
每个用户导入 CSV 行的最大字符数	16000	否	不适用
最大 CSV 文件大小	100 MB	否	不适用
每个 CSV 文件的最大用户数	500,000	否	不适用

Amazon Cognito 身份池（联合身份）资源配额

资源	限额	可调整	最大配额
每个账户的身份池	1000	是	不适用
每个身份池的 Amazon Cognito 用户池提供商数	50	可以	1000
身份池名称的字符长度	128 字节	否	不适用
登录提供商名称的字符长度	2,048 字节	否	不适用
每个身份池的身份	无限制	否	不适用
可以为其指定角色映射的身份提供商数	10	否	不适用
单个列表或查找调用的结果数	60	否	不适用
基于角色的访问控制（RBAC）规则	25	否	不适用

Amazon Cognito Sync 资源配额

资源	限额	可调整	最大配额
每个身份的数据集	20	是	请联系您的账户团队。
每个数据集的记录数	1024	是	请联系您的账户团队。
单个数据集的大小	1 MB	是	请联系您的账户团队。

资源	限额	可调整	最大配额
数据集名称的字符数	128 字节	否	不适用
请求成功后批量发布的等待时间	24 小时	否	不适用

Amazon Cognito 文档历史记录

下表介绍 Amazon Cognito 文档的重要补充部分。我们还会根据您发送的反馈对文档进行频繁的小幅更新。要提交反馈，请找到 Amazon Cognito 文档中任何页面底部的反馈链接。

变更	说明	日期
更新了 Lambda 触发器示例。	更新了自定义电子邮件和短信发送器 Lambda 触发器的示例函数，使其与 Node.js 22.x 兼容。现在，该示例也更易于测试。	2025 年 5 月 19 日
新`prompt`参数。	现在，您可以使用`prompt`参数更好地控制现有托管登录会话的重新身份验证。您也可以将此参数的值传递给第三方提供商。	2025 年 5 月 15 日
M2M 请求的客户端元数据。	现在，您可以在客户端凭据或 machine-to-machine (M2M) 请求中传递客户端元数据。Amazon Cognito 将 M2M 客户端元数据传递给代币生成前 Lambda 触发器。	2025 年 4 月 29 日
刷新代币轮换。	现在，您可以获取新的刷新令牌并使刷新令牌请求中的原始令牌失效。	2025年4月22日
Amazon Cognito 现已在亚太地区（马来西亚）上市。Amazon Web Services 区域	现在，您可以在亚太地区（马来西亚）地区创建 Amazon Cognito 资源。	2025 年 3 月 7 日
机器身份的访问令牌自定义。	令牌生成前 Lambda 触发器现在具有版本三事件，该事件修改了 machine-to-machine	2025 年 3 月 3 日

	(M2M) 授权的客户端凭证授予中的访问令牌声明和范围。	
更新了有关AmazonCognitoPowerUser Amazon 托管策略的信息。	在 Amazon Cognito 用户池高级用户的托 Amazon 管策略中添加了一项 Amazon End User Messaging SMS 操作。	2025 年 2 月 27 日
更新了 OpenID Connect (OIDC) 集成概述。	添加了一张图表，说明了 Amazon Cognito 如何通过 OIDC 身份提供商进行身份验证。	2025 年 2 月 25 日
添加了有关 MFA 逻辑的信息。	添加了一个图表，说明了 Amazon Cognito 如何在运行时将您的用户池多重身份验证 (MFA) 设置应用于用户。	2025 年 2 月 25 日
添加了 Amazon Cognito 用户池安全最佳实践。	添加了一个关于保护机密信息以及以其他方式在用户池配置中遵循安全最佳实践的页面。	2025 年 2 月 25 日
更新了用户池的入门资源。	Amazon Cognito 用户池的入门体验具有新的控制台设计和应用程序选项。	2024 年 11 月 21 日
带有功能计划的新定价模式。	更新了用户池的计费模式。高级安全功能现在是威胁防护。高级安全功能许可证中的组件现已包含在 Essentials 和 Plus 功能计划中。	2024 年 11 月 21 日
新的托管登录功能。	启动托管登录，这是对托管用户界面的更新。	2024 年 11 月 21 日
一种新的身份验证方法和新的身份验证流程。	现在，您可以使用密钥和一次性密码登录 Amazon Cognito 用户池。	2024 年 11 月 21 日

更新了有关 AmazonCognitoUnAuthedIdentitiesSessionPolicy 的信息。	将 Amazon 托管策略中用于缩小未经身份验证的身份范围的 Amazon Key Management Service 操作从内联策略移至托管策略。 Amazon	2024 年 11 月 1 日
添加了 login_hint 参数。	现在，您可以在托管界面、OIDC IdPs 和 Google 的授权请求中添加用户名提示。 IdPs	2024 年 10 月 3 日
电子邮件 MFA 的新高级安全特征。	现在，您可以通过带有高级安全特征的电子邮件消息发送多重身份验证（MFA）代码。	2024 年 9 月 12 日
新内容和页面变更。	修改了标题，删除了不需要的内容，添加了基于场景的介绍，将用户池 OIDC 和托管 UI 端点引用移到了用户池部分。	2024 年 9 月 9 日
更新了有关 AmazonCognitoUnAuthedIdentitiesSessionPolicy 的信息。	用于缩小身份池中未经身份验证的身份的范围的 Amazon 托管策略现在允许 Amazon Location Service。	2024 年 8 月 9 日
通过 Lambda 触发器和增强型威胁检测实现自定义身份验证的新威胁防护。	现在，您可以使用威胁防护分析自定义身份验证登录，并应用自适应身份验证响应。威胁防护现在还会分析登录流量，找出两次尝试之间不合逻辑的地理位置变化。	2024 年 8 月 8 日
新的高级安全特征，用于防止密码重用和用户活动日志导出。	现在，您可以使用 Amazon Cognito 用户池中的高级安全特征导出用户活动日志和设置密码历史记录策略。	2024 年 8 月 6 日

Amazon Cognito 现已在加拿大西部（卡尔加里）和亚太地区（香港） Amazon Web Services 区域推出。	现在，您可以在加拿大西部（卡尔加里）和亚太地区（香港）区域创建 Amazon Cognito 资源。	2024 年 7 月 9 日
改进了应用程序行为的描述以提高安全性	更新了有关高级安全自适应身份验证的设备上下文数据的信息。	2024 年 6 月 10 日
令牌生成前 Lambda 触发器中增加了对复杂对象的支持	现在，您可以将数组和 JSON 对象添加到 ID 令牌和访问令牌声明中。	2024 年 5 月 30 日
更新了有关 Verified Permissions 和 Amazon Cognito 的信息。	Amazon Verified Permissions 现在可以更直接地与 Amazon Cognito 集成。	2024 年 5 月 15 日
经过多区域 Amazon SES 验证的身份。	在某些 Amazon Web Services 区域 没有 Amazon SES 的情况下，Amazon Cognito 用户池在两个远程区域之间进行负载平衡电子邮件。	2024 年 5 月 10 日
添加了有关 M2M 授权和管理成本的信息。	了解如何在 Amazon Cognito 用户池中使用 machine-to-machine (M2M) 用例的客户凭证授权。	2024 年 5 月 9 日
Amazon Cognito 现已在欧洲（西班牙）和亚太地区（海得拉巴） Amazon Web Services 区域推出。	现在，您可以在欧洲（西班牙）和亚太地区（海得拉巴）区域创建 Amazon Cognito 资源。	2024 年 4 月 15 日
Amazon Cognito 现已在亚太地区（墨尔本）上市。 Amazon Web Services 区域	现在，您可以在亚太地区（墨尔本）区域创建 Amazon Cognito 资源。	2024 年 4 月 4 日

在 Flutter 中为 Amazon Cognito 用户池添加了示例 Android 应用程序。	你可以通过上面的 Flutter 示例应用程序为亚马逊 Cognito 构建一款入门级移动应用程序。GitHub	2024 年 4 月 4 日
新的入门内容	有关入门、常见场景、多租户最佳实践以及登录后访问资源的扩展内容。	2024 年 4 月 1 日
Amazon Cognito 现已在欧洲（苏黎世）上市。 Amazon Web Services 区域	您现在可以在欧洲（苏黎世）区域创建 Amazon Cognito 资源。	2024 年 3 月 14 日
Amazon Cognito 现已在中东（阿联酋）上市。 Amazon Web Services 区域	您现在可以在中东（阿联酋）区域创建 Amazon Cognito 资源。	2024 年 3 月 8 日
新的 SAML 特征和改进的内容。	现在，您可以签署 SAML 请求、加密 SAML 响应以及设置 IdP 发起的 SAML SSO。	2024 年 2 月 1 日
可增加配额。	现在，您可以为 Amazon Cognito 请求速率配额购买更多容量。	2024 年 1 月 25 日
Amazon Cognito 身份池支持 Service Quotas 中的请求速率。	现在，您可以监控 Amazon Cognito 身份池的 requests-per-second (RPS) 配额，并在 Service Quotas 控制台中请求增加配额。	2023 年 12 月 19 日
添加了自定义访问令牌内容的新功能。	现在，您可以在用户池访问令牌中添加、修改和删除声明和范围。	2023 年 12 月 12 日

改进了有关应用程序客户端和 OAuth 范围的内容。	澄清对 特定于应用程序的应用程序客户端设置 和 作用域、M2M 以及 APIs 带资源服务器的场景 所做的编辑和更正。删除了旧版控制台指令。	2023 年 11 月 14 日
改进了有关设备和设备身份验证的内容。	有关使用设备密钥和设备 SRP 身份验证的新内容。	2023 年 10 月 18 日
更新了 Amazon Web Services Management Console 指南。	删除了用户池控制台参考以及相关主题中重新分发的主题，并为 Amazon Cognito 控制台中基于选项卡的整理方式添加了指南。	2023 年 8 月 30 日
不再强调直接访问 LOGIN 端点。	添加了用户池登录端点的直观概述，并强调了从对端点授权开始身份验证。	2023 年 8 月 30 日
Amazon Cognito 现已在亚太地区（大阪）和以色列（特拉维夫）上市。 Amazon Web Services 区域	现在，您可以在亚太地区（大阪）和以色列（特拉维夫）区域中创建 Amazon Cognito 资源。	2023 年 8 月 30 日
介绍了有关使用 Amazon Verified Permissions 为 Amazon Cognito 授权的信息。	在您的应用程序中，您可以调用 Verified Permissions API 从中央机构生成访问决策。	2023 年 8 月 1 日
在 Amazon CloudWatch Logs 中添加了一项用于记录用户池详细用户活动的新功能。	现在，您可以将电子邮件和 SMS 消息传送错误 CloudWatch 记录到日志组中。	2023 年 8 月 1 日
更新了有关身份池访客用户的 Amazon 托管策略的信息。	身份池访客用户的权限范围现在包括内联会话策略和 Amazon 托管会话策略。	2023 年 5 月 16 日

Amazon Cognito 身份池的内容改进和新的控制台说明。	添加了新的控制台演练以反映新的控制台体验，改进了身份池的代码集成详细信息。	2023 年 5 月 16 日
对服务主页和用户池主页的添加和改进。	更新了 Amazon Cognito 和用户池的概述页面。	2023 年 5 月 16 日
对用户池令牌文档进行了总体改进。	更新了示例令牌，添加了有关验证令牌的新信息。	2023 年 2 月 16 日
您现在可以在 Amazon CloudTrail中记录 Amazon Cognito 身份池数据事件。	CloudTrail 支持在记录数据事件的跟踪中选择 Amazon Cognito 身份池的大容量 API 操作。	2023 年 2 月 15 日
更新了 Lambda 触发器示例和描述。	Lambda 触发器示例已更新至 JavaScript 版本 3。您现在可以直接将 Lambda 触发器与 API 操作相关。	2023 年 1 月 31 日
Amazon Cognito 身份池将 Amazon 托管策略应用于未经身份验证的会话。	使用增强型流程进行身份验证的身份池用户现在可以对其会话应用额外的 Amazon 托管策略。	2023 年 1 月 31 日
添加了代码示例。	本指南现在包含各种编程语言的 Amazon Cognito 应用程序的示例代码。	2023 年 1 月 23 日
添加了有关 API 模型和使用 Amazon Cognito 用户池进行身份验证的信息。	Amazon Cognito 用户池具有多种用于请求授权的 API 接口和格式。	2022 年 12 月 15 日
亚马逊 Cognito 现已在欧洲（米兰）上市。 Amazon Web Services 区域	您现在可以在欧洲地区（米兰）区域创建 Amazon Cognito 用户池。	2022 年 12 月 6 日

添加了有关用户池删除保护的信息。	当你使用创建新的用户池时 Amazon Web Services Management Console，默认情况下会保护该用户池不被删除。	2022 年 10 月 20 日
添加了托管 UI 的用户指南以及有关托管 UI 中的 TOTP MFA 的信息。	您的用户现在可以在 Amazon Cognito 托管 UI 中注册 TOTP MFA 设备。现在，您可以预览默认的托管 UI。	2022 年 9 月 8 日
添加了有关 Amazon WAF 和 Amazon Cognito 的信息。	现在，您可以将 Amazon WAF 网页 ACL 与 Amazon Cognito 用户池相关联。	2022 年 8 月 3 日
添加了更多示例 Amazon CloudTrail 事件。	Amazon Cognito 现在会将联合身份验证和托管 UI 请求记录到您的跟踪。	2022 年 6 月 15 日
添加了有关两步属性验证的信息。	现在，您可以选择您的用户是否必须验证新的电子邮件地址或电话号码，然后才能使用该地址或电话号码登录。	2022 年 6 月 9 日
更新了联合身份验证文档。新的 IP 地址传播特征。	更新了设置用户池社交 IdPs的演练。添加了有关联合用户配置文件和属性映射的信息。添加了有关设备指纹的新信息，以提高安全性。	2022 年 5 月 31 日
使联合用户登录，而无需与托管 UI 交互	添加了一个关于如何收藏应用程序的新页面，以便 Amazon Cognito 静默地将用户定向到联合身份验证登录。	2022 年 5 月 29 日

Amazon Cognito 用户池的区域内 SMS 和电子邮件消息	现在，您可以在用户池中使用亚马逊简单通知服务发送短信，使用亚马逊简单电子邮件服务发送电子邮件。 Amazon Web Services 区域	2022 年 3 月 14 日
配额页面更新	添加并澄清了资源和请求速率配额。	2022 年 1 月 10 日
全新 Amazon Cognito 用户池控制台体验	更新了在更新后的 Amazon Cognito 控制台中创建和管理用户池的说明。	2021 年 11 月 18 日
RevokeToken API 和撤销终端节点	您可以使用该 RevokeToken 操作来撤消用户的刷新令牌。	2021 年 6 月 10 日
多租户最佳实践	添加了多租户应用程序的最佳实践。	2021 年 3 月 4 日
访问控制属性	Amazon Cognito 身份池提供访问控制 (AFAC) 属性，作为客户向用户授予资源访问权限的一种方式。 Amazon 授权可以根据用户用来与 Amazon Cognito 联合的身份提供商提供的用户属性执行。	2021 年 1 月 15 日
自定义 SMS 发件人 Lambda 触发器和自定义电子邮件发件人 Lambda 触发器	您可以利用自定义 SMS 发件人 Lambda 触发器和自定义电子邮件发件人 Lambda 触发器使第三方提供商从 Lambda 函数代码中向用户发送电子邮件和 SMS 通知。	2020 年 11 月 30 日
Amazon Cognito 令牌更新	在访问令牌、ID 令牌和刷新令牌中添加了更新的过期信息。	2020 年 10 月 29 日

Amazon Cognito Service Quotas	Service Quotas 可用于 Amazon Cognito 类别配额。您可以使用 Service Quotas 控制台查看配额使用情况、请求增加配额以及创建 CloudWatch 警报以监控您的配额使用情况。作为此次变更的一部分，Amazon Cognito 用户池的可用 CloudWatch指标部分已更新，以反映新信息。新的栏目名称为：跟踪和 Service Quotas 中的配额 CloudWatch 和使用情况	2020 年 10 月 29 日
Amazon Cognito 配额分类	配额分类可用于帮助您监控配额使用情况并请求增加配额。配额根据常见使用案例分为不同的类别。	2020 年 8 月 17 日
美国政府支持亚马逊 Cognito Amazon CLoud	Amazon GovCloud （美国）地区现已支持 Amazon Cognito。	2020 年 5 月 13 日
Amazon Cognito Pinpoint 文档更新	添加了新的服务相关角色。更新了有关“将 Amazon Pinpoint 分析与 Amazon Cognito 用户池结合使用”的说明。	2020 年 5 月 13 日
新 Amazon Cognito 专用安全性章节	“安全” 章节可以帮助您的组织获得有关 Amazon 服务的内置和可配置安全性的深入信息。我们的新章节提供有关云及云中安全性的信息。	2020 年 4 月 30 日
Amazon Cognito 身份池现在支持“通过 Apple 登录”	“通过 Apple 登录”功能现已在所有运营 Amazon Cognito 的区域推出，cn-north-1 region 除外。	2020 年 4 月 7 日

新的 Facebook API 版本控制	在 Facebook API 中添加了版本选择。	2020 年 4 月 3 日
用户名不区分大小写更新	添加了关于在创建用户池之前启用用户名不区分大小写的建议。	2020 年 2 月 11 日
有关的新信息 Amazon Amplify	添加了有关使用和库将 Amazon Cognito 与您的网络或移动应用程序集成的信息 Amazon Amplify SDKs 。删除了之前有关使用 Amazon Cognito SDKs 的信息。Amazon Amplify	2019 年 11 月 22 日
用户池触发器的新属性	现在，Amazon Cognito 在事件信息中包含一个`clientMetadata` 参数，该参数会传递给大多数用户池触发器的 Amazon Lambda 函数。您可以使用此参数来通过其他数据强化自定义身份验证工作流。	2019 年 10 月 4 日
更新限制	ListUsers API 操作的限制已更新。	2019 年 6 月 25 日
新限制	用户池中的软性限制现在包含对用户数量的限制。	2019 年 6 月 17 日
Amazon Cognito 用户池的 Amazon SES 电子邮件设置	您可以配置用户池，以便 Amazon Cognito 使用您的 Amazon SES 配置向用户发送电子邮件。此设置允许 Amazon Cognito 以比其他可能的方式更高的送达量发送电子邮件。	2019 年 4 月 8 日

标记支持	添加了有关标记 Amazon Cognito 资源的信息。	2019 年 3 月 26 日
更改自定义域的证书	如果您使用自定义域托管 Amazon Cognito 托管 UI，则可以根据需要更改此域的 SSL 证书。	2018 年 12 月 19 日
新限制	添加了针对每个用户可以属于的最大组数的新限制。	2018 年 12 月 14 日
已更新限制	更新了用户池的软限制。	2018 年 12 月 11 日
用于验证电子邮件地址和电话号码的文档更新	添加了有关配置用户池以在用户注册您的应用程序时要求进行电子邮件或电话验证的信息。	2018 年 11 月 20 日
用于测试电子邮件的文档更新	添加了有关在测试应用程序时从 Amazon Cognito 启动电子邮件传送的指导信息。	2018 年 11 月 13 日
Amazon Cognito 高级安全性	新增的安全功能使开发人员能够保护他们的应用程序和用户免受恶意的自动程序攻击，确保用户账户拒绝已泄露的凭证，并根据计算的登录尝试风险自动调整登录所需的难度。	2018 年 6 月 14 日
自定义域用于 Amazon Cognito 托管 UI	允许开发人员将自己的完全自定义域用于 Amazon Cognito 用户池中的托管 UI。	2018 年 4 月 6 日
Amazon Cognito 用户池 OIDC 身份提供商	已添加“通过 OpenID Connect（OIDC）身份提供商（如 Salesforce 或 Ping Identity）登录用户池”。	2018 年 5 月 17 日

Amazon Cognito Lambda 迁移触发器	添加了介绍 Lambda 迁移触发器功能的页面	2018 年 4 月 8 日
Amazon Cognito 开发人员指南更新	添加了顶层内容“什么是 Amazon Cognito”和“Amazon Cognito 入门”。还添加了常见场景并重新组织了用户池 TOC。添加了新的“Amazon Cognito 用户池入门”部分。	2018 年 4 月 6 日
Amazon Cognito 高级安全性测试版	添加的新安全功能，可让开发人员保护应用和用户远离恶意自动程序，确保已在 Internet 上泄露的凭证无法登录用户账户，并根据登录尝试计算出风险，据此自动调整登录所需的难度。	2017 年 11 月 28 日
Amazon Pinpoint 集成	添加了使用 Amazon Pinpoint 为您的 Amazon Cognito 用户池应用程序提供分析并丰富 Amazon Pinpoint 活动用户数据的功能。	2017 年 9 月 26 日
Amazon Cognito 用户池的联合身份验证和内置应用程序 UI 特征	添加了允许用户通过 Facebook、Google、Login with Amazon 或 SAML 身份提供商登录用户池的功能。添加了可自定义的内置应用程序 UI 和带有自定义声明的 OAuth 2.0 支持。	2017 年 8 月 10 日
HIPAA 和 PCI 合规性相关的特征变更	添加了允许用户使用电话号码或电子邮件地址作为用户名的功能。	2017 年 6 月 7 日

用户组和基于角色的访问控制特征	添加了创建和管理用户组的管理功能。管理员可以根据组成员资格和管理员创建的规则将 IAM 角色分配给用户。	2016 年 12 月 15 日
文档更新	更新了显示如何在用户池中使用 Amazon Lambda 触发器的示例。	2016 年 11 月 27 日
文档更新	更新了 iOS 代码示例。	2016 年 11 月 18 日
文档更新	添加了有关用户账户的确认流程的信息。	2016 年 11 月 9 日
创建用户账户特征	添加了通过 Amazon Cognito 控制台和 API 创建用户账户的管理功能。	2016 年 10 月 6 日
用户导入特征	添加了 Cognito 用户池的批量导入功能。使用此功能将用户从现有身份提供商迁移到 Amazon Cognito 用户池。	2016 年 9 月 1 日
Cognito 用户池正式发布	添加了 Cognito 用户池功能。借助此功能，使用用户池创建和维护用户目录，并将注册信息和登录信息添加到移动应用程序或 Web 应用程序中。	2016 年 7 月 28 日
SAML 支持	使用身份提供商通过安全断言标记语言 2.0（SAML 2.0）添加了对身份验证的支持。	2016 年 6 月 23 日
CloudTrail 整合	增加了与的集成 Amazon CloudTrail。	2016 年 2 月 18 日

事件与 Lambda 的集成	使您能够在 Amazon Cognito 中执行 Amazon Lambda 函数以响应 Amazon Cognito 中的重要事件。	2015 年 4 月 9 日
Amazon Kinesis 的数据流	提供了对数据流的控制和了解。	2015 年 3 月 4 日
OpenID Connect 支持	启用了对 OpenID Connect 提供商的支持。	2014 年 11 月 23 日
推送同步	启用了对无提示推送同步的支持	2014 年 11 月 6 日
添加了对经过开发人员验证的身份的支持	使拥有自己的身份验证和身份管理系统的开发人员被视为 Amazon Cognito 中的身份提供商。	2014 年 9 月 29 日
Amazon Cognito 正式发布		2014 年 7 月 10 日

本文属于机器翻译版本。若本译文内容与英语原文存在差异，则一律以英文原文为准。